MULTIPLE SEQUENCE ALIGNMENT OF BIOLOGY
BASED ON EVOLUTIONARY ALGORITHMS

BIOLOGY



进化算法在生物多序列比对中的应用

龙海侠　李满枝　王洪涛　付海艳 ⊙著

清华大学出版社

进化算法在生物多序列比对中的应用

龙海侠　李满枝
王洪涛　付海艳　著

清华大学出版社
北　京

内 容 简 介

本书全面系统地介绍了进化算法在生物多序列比对中的应用，根据内容的分类，分为“多序列比对基础篇”“多序列比对模拟篇”和“多序列比对参数篇”三个模块。首先介绍生物多序列比对的基础知识，包括多序列比对的基本概念、原理、方法、常用数据库、常用工具和应用等内容，并介绍进化算法和最优化理论的基础知识，以及遗传算法、粒子群优化算法和量子粒子群优化算法的优化过程及收敛性分析，为进行多序列比对的模拟提供理论基础；然后详细介绍各进化算法模拟多序列比对的过程与结果；最后对于多序列比对最重要的目标函数参数进行建模与分析。本书具有系统性强、可读性强、可操作性强等特点。

图书在版编目(CIP)数据

进化算法在生物多序列比对中的应用/龙海侠 等著. —北京：清华大学出版社，2017
ISBN 978-7-302-46806-6

I. ①进…　II. ①龙…　III. ①最优化算法—应用—生物分析—研究　IV. ①O242.23　②Q-33

中国版本图书馆 CIP 数据核字(2017)第 054017 号

责任编辑： 王　定　程　琪
封面设计： 周晓亮
版式设计： 思创景点
责任校对： 牛艳敏
责任印制：

出版发行： 清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　邮　　编：100084
社 总 机：010-62770175　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销： 全国新华书店
开　　本： 148mm×210mm　印　　张：9　字　　数：251 千字
版　　次： 2017 年 5 月第 1 版　印　　次：2017 年 5 月第 1 次印刷
印　　数： 1～1000
定　　价： 98.00 元

产品编号：

作 者 简 介



龙海侠 1980年生，2007年获江南大学计算机软件与理论硕士学位，2010年获江南大学轻工信息技术与工程博士学位，现就职于海南师范大学信息科学技术学院，副教授。研究方向：群体智能算法、进化算法、生物信息。硕士期间从事群体智能算法和进化算法的研究及其在聚类、图像分割上的应用研究；博士期间从事生物信息的研究，重点研究多序列比对和培养基的优化；近5年从事深度学习算法和生物信息的研究。已出版教材1部、专著1部，发表论文30余篇，主持省级课题2项，作为第一完成人获得省级奖励2项。



李满枝 1979年生，2004年6月获西北工业大学计算数学专业理学硕士学位，现就职于海南师范大学数学与统计学院，副教授。主要研究方向：生物信息学、计算机数值模拟、算法构造等。硕士期间从事基于蒙特卡罗方法的计算机模拟，近5年从事生物信息中的蛋白质功能预测研究。已在国内外核心期刊及学术会议上发表多篇论文，出版专著1部，并作为主要成员参与省级和国家级自然科学项目多项，现主持海南省自然科学基金“生物多序列比对的遗传算法模拟及改进”。



王洪涛　1978年生，2008年6月获海南师范大学应用数学专业理学硕士学位，现就职于海南师范大学数学与统计学院，副教授。主要研究方向：计算机数值模拟、算法构造、数学建模等。在国内外核心期刊及学术会议上发表多篇论文，出版专著1部，并作为主要成员参与海南省自然科学基金项目多项，目前是海南省自然科学基金“生物多序列比对的遗传算法模拟及改进”的第二参与人。



付海艳　1978年生，2002年获山东大学人工智能与模式识别硕士学位，2009年获山东大学系统理论博士学位，现就职于海南师范大学信息科学技术学院，教授。研究方向：评价理论与方法、决策理论与方法、不确定信息处理。硕士期间从事基于模糊集理论的评价方法和决策方法的研究，博士期间从事基于粗糙集理论和模糊集理论的不确定信息处理，近5年从事数据挖掘算法的研究。已出版教材2部、专著1部，发表论文30余篇，主持国家级课题1项、省级课题6项，作为第一完成人获得省级奖励2项。

前　　言

随着人类基因组计划的实施和科技的发展，生物学数据呈爆炸式增长，这些海量的生物学数据必须通过生物信息学手段进行收集、分析和整理后，才能成为有用的信息。而如何有效分析和处理这些大型序列数据(即序列分析)成为生物信息学的首要任务。序列比对是生物序列分析的主要方法，也是生物信息学中挑战性的问题之一。序列比对在序列装配、序列注释、基因和蛋白质的结构和功能预测以及系统发育和进化分析等方面均有广泛应用，因此对它的研究一直以来都是热点。

进化算法是一类借鉴生物界自然选择和自然遗传机制的随机搜索算法，主要包括遗传算法(genetic algorithm，GA)、遗传规划(genetic programming，GP)、进化策略(evolutionary strategies，ES)、进化规划(evolutionary programming，EP)、粒子群优化(particle swarm optimization，PSO)算法以及近年出现的量子粒子群优化(quantum-behaved particle swarm optimization，QPSO)算法，它们通过一系列的进化算子和进化方程，寻找问题的最优解。本书把上述的进化算法及其改进的进化算法，结合数学模型，用于解决生物多序列比对问题。

全书正文各章节结构如下图所示，共分为“多序列比对基础篇”“多序列比对模拟篇”和“多序列比对参数篇”三个模块。

篇	章
上篇 多序列比对基础篇	第1章 生物多序列比对
	第2章 进化算法和最优化理论
	第3章 遗传算法、粒子群优化算法和量子粒子群优化算法
中篇 多序列比对模拟篇	第4章 遗传算法在多序列比对中的应用
	第5章 QPSO算法在多序列比对中的应用
	第6章 基于隐马尔可夫模型和QPSO算法的多序列比对
	第7章 多序列比对的并行计算
下篇 多序列比对参数篇	第8章 多序列比对的参数研究

“多序列比对基础篇”(第 1 章～第 3 章)介绍生物多序列比对的基础知识，包括多序列比对的基本概念、原理、方法、常用数据库、常用工具和应用等内容，并介绍进化算法和最优化理论的基础知识，以及遗传算法、粒子群优化算法和量子粒子群优化算法的优化过程及收敛性分析，为进行多序列比对的模拟提供理论基础。

“多序列比对模拟篇”(第 4 章～第 7 章)是本书的核心部分，主要内容概括如下：

(1) 应用基本遗传算法及其改进的遗传算法进行多序列比对。基本遗传算法(GA)是通过对进化过程中的种群反复进行选择、交叉、变异操作来模拟自然界中种群的演变过程，直到满足一定性能要求才结束计算，它本身的结构决定了它可以用在多序列比对上。遗传

算法可以有效地解决生物多序列比对问题，但是遗传算法高度依赖于初始种群，好的初始种群方可以得到好的结果。为提高计算效率，提高比对质量，可从遗传算法最关键的组成部分入手，通过优化初始种群的质量，达到改进算法的目的。另外，又针对遗传算法最基本的交叉算子，设计了保优和选择混合的交叉操作后处理方法cross4to2。该方法不但服从保优原则，而且又再一次经过选择操作的精英保留过程，使得最优秀的个体进入下一代。这种处理将算法的整体搜索能力和局部搜索能力大大提高。通过与经典 CLUSTAL 算法的比较，验证了该算法的有效性。

(2) 使用二进制的 PSO 算法和二进制的 QPSO 算法进行多序列的比对。为了避免算法的早熟，在算法中还加入了变异算子。首先对群体中的个体进行编码，然后根据目标函数值(通常为序列的得分函数)找出空位的最优位置，使序列比对的结果最优，确定序列的相似性以至于同源性。

(3) 使用 QPSO 算法和改进的 QPSO 算法，结合隐马尔可夫模型(HMM)进行多序列的比对。这主要涉及两个过程：优化过程和比对过程。优化过程主要研究剖面 HMM 模型参数的训练过程，获得较优模型。前面已经提及现有的训练算法通常会陷入局部最优，因此研究全局优化算法对模型进行训练极其重要。用并行的群体智能优化算法优化剖面 HMM 时，优化的主要对象是转移概率和符号发出概率，优化对象的编码方式以及参数的个数将会影响比对的速度，优化过程中算法的全局收敛性将会影响到比对的准确度。比对过程主要研究比对算法的实现过程，获得比对结果。当使用 HMM 进行多序列比对时，每条序列从开始到结束通过这些状态穿越模型，在这些待比对序列中进行空位字符“－”的插入和删除操作，得到一个多序列比对结果的矩阵。但应确保在比对结果中有尽可能多的列

由相同的非空字符组成，同时在由不同字符组成的列中某一个或某几个非空字符的数目尽可能多，以便发现不同序列之间的相似部分，进而推断它们在功能和结构上的相似性。

(4) 多序列比对的并行计算。随着计算机科学技术在第三代测序技术以及基因组拼接技术方面的不断发展，生物信息领域获得了越来越多的长基因组序列数据，长序列比对成为急需解决的问题。传统的算法对内存空间的庞大需求以及漫长的运行时间已经无法满足对这种大规模数据的处理，因此长序列比对的并行计算成为研究的一个热点问题。通常的并行模式有：基于“分而治之”策略，结合并行计算的长序列首尾分段并行比对算法；基于“粗细粒度”的并行数据并行算法。

多序列比对是生物信息学的一个重要研究内容，比对结果高度依赖于目标函数和比对工具的参数设置，包括空位罚分(GOP 和 GEP)以及替换矩阵。“多序列比对参数篇”(第 8 章)主要做了两方面的工作：

(1) 研究 SP(sum-of-pair)目标函数，提出确定各参数最优值的理论依据，给出替换矩阵判断公式和最佳空位罚分取值公式，结合待测序列信息得出与之相符的一组最优参数，从而得到更好的比对结果。通过与精度较高的多序列比对工具 MAFFT、CLUSTALW 的比较，结合 BAliBASE2.0 数据库进行实例验证，结果表明，根据公式得出的参数可以得到比默认参数更优的比对结果，而且本书公式优化了多序列比对结果，具有可行性和高效性。

(2) 基于 BAliBASE3.0 数据库，应用 MAFFT 工具(MAFFT-7.220-WIN64 version)进行多序列比对，得出替换矩阵和空位罚分的最优参数组合，从而得到更好的比对结果。实验结果表明，通过与 MAFFT (MAFFT-7.220-WIN64 version)、CLUSTALW (CLUSTALW-2.1-WIN)

的默认参数比较，根据本研究得出的最优参数组合可以得到比默认参数更优的比对结果，而且研究结果给出的最优参数组合优化了多序列比对结果。

本书是由多人编撰完成的，编写分工如下：第 5 章、第 6 章和附录 I～J 由龙海侠编撰完成，共计 9 万字；第 4 章、第 8 章和附录 A～H 由李满枝编撰完成，共计 9 万字；第 1 章、第 7 章由王洪涛编撰完成，共计 8.5 万字；第 2 章和第 3 章由付海艳编撰完成，共计 8.5 万字。全书由龙海侠和李满枝统稿和修改。本书的出版获海南师范大学学术著作出版资助项目、海南省自然科学基金项目(20151003，614235)、国家自然科学基金(71461008)、海南师范大学数学与统计学院“计算数学”重点学科和信息科学技术学院“计算机科学与技术”一级学科的资助，特此表示感谢。

本书可作为生物信息学、计算生物学、计算机和计算数学等专业本科生或研究生的教材或学习参考书，也可作为相关研究人员的研究参考书。由于我们的专业知识与工作背景的限制，书中还有很多错误或不足之处，敬请希望读者批评指正。

龙海侠　李满枝

2017 年 1 月于海南师范大学

目　　录

上篇　多序列比对基础篇

中篇 多序列比对模拟篇

下篇 多序列比对参数篇

上　篇

多序列比对基础篇

第 1 章　生物多序列比对

1.1　生物信息学

1.1.1　生物信息学的起源

自从 1990 年美国启动人类基因组计划以来，人与模式生物基因组的测序工作进展极为迅速。迄今已完成了约 40 多种生物的全基因组测序工作，人基因组约 3×10^9 个碱基对的测序工作也接近完成。至 2000 年 6 月 26 日，被誉为生命“阿波罗计划”的人类基因组计划，经过美、英、日、法、德和中国科学家的艰苦努力，终于完成了工作草图，这是人类科学史上又一个里程碑式的事件，它预示着完成人类基因组计划已经指日可待。截至目前，仅登录在美国 GenBank 数据库中的 DNA 序列总量已超过 70 亿个碱基对。在人类基因组计划进行过程中所积累起来的技术和经验，使得其他生物基因组的测序工作可以完成得更快捷。可以预计，今后 DNA 序列数据的增长将更为惊人。生物学数据的积累并不仅仅表现在 DNA 序列方面，与其同步的还有蛋白质的一级结构，即氨基酸序列的增长。此外，迄今为止，已有 10 000 多种蛋白质的空间结构以不同的分辨率被测定。基于 cDNA 序列测序所建立起来的 EST 数据库，其记录已达数百万条。在这些数据基础上派生、整理出来的数据库已达 500 余个。这一切构成了一个生物学数据的海洋。可以打一个比方来说明这些数据的规模。有人估计，人类(包括已经去世的和仍然在世的)所说过的话的信息总量约为 5EB($1EB=10^{18}B$)，而如今生物学数据信息总量已接近甚至超过此数量级。这种科学数据的急速和海量积累，在

人类的科学研究历史中是空前的。

数据并不等于信息和知识，但却是信息和知识的源泉，关键在于如何挖掘它们。与正在以指数方式增长的生物学数据相比，人类相关知识的增长(粗略地用每年发表的生物、医学论文数来代表)却十分缓慢。一方面是巨量的数据；另一方面是我们在医学、药物、农业和环保等方面对新知识的渴求，这些新知识将帮助人们改善其生存环境和提高生活质量。这就构成了一个极大的矛盾。这个矛盾就催生了一门新兴的交叉科学，这就是生物信息学。

1.1.2　生物信息学的概念

美国人类基因组计划实施五年后的总结报告中，对生物信息学做了以下定义：生物信息学是一门交叉科学，它包含了生物信息的获取、处理、存储、分发、分析和解释等在内的所有方面，它综合运用数学、计算机科学和生物学的各种工具，来阐明和理解大量数据所包含的生物学意义。生物信息学这一名词的出现仅仅是几年前的事情，但是计算生物学这一名词的出现要早得多。鉴于这两门学科之间并没有或难以界定严格的分界线，在这里统称为生物信息学。它是当今生命科学和自然科学的重大前沿领域之一，同时也是 21 世纪自然科学的核心领域之一。其研究重点主要体现在基因组学(genomics)和蛋白组学(proteomics)两方面，具体说就是从核酸和蛋白质序列出发，分析序列中表达的结构功能的生物信息。

1.1.3　生物信息学的主要研究内容

生物信息学主要包括以下几个主要研究领域，但是限于篇幅，这里仅列出其名称并只做简单介绍。

1. 序列比对(alignment)

基本问题是比较两个或两个以上符号序列的相似性或不相似性。

序列比对是生物信息学的基础，非常重要。两个序列的比对有较成熟的动态规划算法，以及在此基础上编写的比对软件包—— BLAST 和 FASTA，可以免费下载使用。这些软件在数据库查询和搜索中有重要的应用。有时两个序列总体并不很相似，但某些局部片断相似性很高。Smith-Waterman 算法是解决局部比对的好算法，缺点是速度较慢。两个以上序列的多重序列比对目前还缺乏快速而又十分有效的算法。

2. 蛋白质三级结构比对

蛋白质三级结构比对是生物信息学的重要研究领域。蛋白质的功能由蛋白质的三级结构决定，蛋白质三维空间结构的相似性比较是分析蛋白质结构和功能的重要手段，因此比较蛋白质的三级结构可以了解它们之间的相互作用和进化关系。

研究蛋白质的结构意义重大，分析蛋白质结构、功能及其关系是蛋白质组计划中的一个重要组成部分。研究蛋白质结构有助于了解蛋白质的功能，了解蛋白质如何行使其生物功能，认识蛋白质与蛋白质或其他分子之间的相互作用，这无论是对于生物学还是对于医学和药学都是非常重要的。对于未知功能或者新发现的蛋白质分子，通过结构分析可以进行功能注释、指导设计进行功能确认的生物学实验。通过分析蛋白质的结构确认功能单位或者结构域，可以为遗传操作提供目标，为设计新的蛋白质或改造已有蛋白质提供可靠的依据，同时为新的药物分子设计提供合理的靶分子结构。

3. 蛋白质二级和三级结构预测

从方法上来看，有演绎法和归纳法两种途径。前者主要是从一些基本原理或假设出发来预测和研究蛋白质的结构和折叠过程，分子力学和分子动力学属这一范畴。后者主要是从观察和总结已知结构的蛋白质结构规律出发来预测未知蛋白质的结构，同源模建和指认(threading)方法属于这一范畴。虽然经过 30 余年的努力，蛋白质结构预测研究现状还远远不能满足实际需要。

4. 计算机辅助基因识别(仅指蛋白质编码基因)

基本问题是给定基因组序列后，正确识别基因的范围和在基因组序列中的精确位置。这是最重要的课题之一，而且越来越重要。经过 20 余年的努力，提出了数十种算法，有 10 种左右重要的算法和相应软件(网上提供免费服务)。原核生物计算机辅助基因识别相对容易些，结果好一些。从具有较多内含子的真核生物基因组序列中正确识别出起始密码子、剪切位点和终止密码子，是一个相当困难的问题，研究现状不能令人满意，仍有大量的工作要做。

5. 非编码区分析和 DNA 语言研究

在人类基因组中，编码部分只占总序列的 3%～5%，其他通常称为“垃圾”DNA，其实一点也不是“垃圾”，只是暂时还不知道其重要的功能。分析非编码区 DNA 序列需要大胆的想象与崭新的研究思路和方法。DNA 序列作为一种遗传语言，不仅体现在编码序列之中，而且隐含在非编码序列之中。

6. 分子进化和比较基因组学

早期的工作主要是利用不同物种中同一种基因序列的异同来研究生物的进化，构建进化树。既可以用 DNA 序列也可以用其编码的氨基酸序列来做，甚至可通过相关蛋白质的结构比对来研究分子进化。以上研究已经积累了大量的工作。近年来由于较多模式生物基因组测序任务的完成，为从整个基因组的角度来研究分子进化提供了条件。可以设想，比较两个或多个完整基因组这一工作需要新的思路和方法，当然也渴望得到更丰硕的成果。这方面可做的工作是很多的。

7. 序列重叠群(contigs)装配

一般来说，根据现行的测序技术，每次反应只能测出 500 个或更多一些碱基对的序列，这就有一个把大量的较短的序列全体

构成重叠群，逐步把它们拼接起来形成序列更长的重叠群，直至得到完整序列的过程，称为重叠群装配。拼接 EST 数据以发现全长新基因也有类似的问题。已经证明，这是一个 NP 完备性算法问题。

8. 遗传密码的起源

遗传密码为什么是现在这样的？这一直是一个谜。一种最简单的理论认为，密码子与氨基酸之间的关系是生物进化历史上一次偶然的事件造成的，并被固定在现代生物最后的共同祖先里，一直延续至今。不同于这种“冻结”理论，有人曾分别提出过选择优化、化学和历史等三种学说来解释遗传密码。随着各种生物基因组测序任务的完成，为研究遗传密码的起源和检验上述理论的真伪提供了新的素材。

9. 基于结构的药物设计

人类基因组计划的目的之一在于阐明人的约 10 万种蛋白质的结构、功能、相互作用以及与各种人类疾病之间的关系，寻求各种治疗和预防方法，包括药物治疗。基于生物大分子结构的药物设计是生物信息学中的极为重要的研究领域。为了抑制某些酶或蛋白质的活性，在已知其三级结构的基础上，可以利用分子对接算法，在计算机上设计抑制剂分子，作为候选药物。这种发现新药物的方法有强大的生命力，也有着巨大的经济效益。

10. 代谢网络的分析

代谢网络涉及生化反应途径、基因调控及信号转导过程(蛋白质间的作用)等。后基因组时代将研究大规模网络的生命过程，称为“网络生物学”研究。与代谢分析直接相关的便是系统生物学研究，它将是后基因组时代最为突出的研究方向。

11. 其他

如基因表达谱分析、基因芯片设计和蛋白质组学数据分析等，逐渐成为生物信息学中新兴的重要研究领域。

1.2　序列比对的概念及其发展历史

1.2.1　序列比对的提出与基本概念

随着人类基因组计划的实施和科技的发展，生物学数据呈爆炸式增长，这些海量的生物学数据必须通过生物信息学手段进行收集、分析和整理后，才能成为有用的信息。而如何有效分析和处理这些大型序列数据(即序列分析)成为生物信息学的首要任务。序列比对是生物序列分析的主要方法，也是生物信息学中挑战性的问题之一。序列比对在序列装配、序列注释、基因和蛋白质的结构和功能预测以及系统发育和进化分析等方面均有广泛应用，因此对它的研究一直以来都是热点。

序列比对就是在两个或更多序列的相同区域寻找最大相似性的任务，其基本思想是找出检测序列和目标序列的相似性。比对过程中需要在待比对序列中引入空位，以表示插入或删除。根据需要比对的序列个数，序列比对分为双序列比对和多序列比对。

序列比对问题就是通过向待测序列中插入空格，使得每条序列的长度一样，并尽可能保证每列的字符具有最大的相似性。随着参与比对序列的条数增加，比对的难度及复杂度急剧增加。

1.2.2　序列比对的目的和意义

当所考察的序列不同时，保守的残基往往是维持稳定结构或生物学功能的关键残基。多序列比对可以揭示关于蛋白质结构和功能

的许多线索。

序列比对的目的是发现相似的序列，得到保守的区域，寻找序列之间的功能、结构或进化上的关系。

序列比对的意义如下：

(1) 用于描述一组序列之间的相似性关系，以便了解一个基因家族的基本特征，寻找 motif、保守区域等。

(2) 用于描述一个同源基因之间的亲缘关系的远近，应用到分子进化分析中。

(3) 定量估计序列间的关系，并由此推断它们在进化中的亲缘关系，可以通过计算完全匹配的残基数目或计算完全匹配残基和相似残基的数目得到这种定量关系。该方法还可以用来评估比对质量。

(4) 对于系统发育，相等的残基相当于具有共同的进化祖先；对于结构生物学，相等的残基与一组蛋白质中同源折叠的类似位置相关；对于分子生物学，相等的残基在其相应的蛋白质有同类功能作用。在每一种情况下，比对提供了潜在的演化、结构，或简洁直观地表达蛋白质家族功能限制表征的鸟瞰视图。

(5) 如果是对多个蛋白质或核酸同时进行比较分析，就有可能寻找到这些有进化关系的序列之间共同的保守区域、位点和 profile，从而就能够探索到导致它们产生共同功能的序列模式。

(6) 序列比对在预测和治疗疾病方面有着非常重要的应用，如白血病。大量的试验数据和临床数据表明，DNA 序列中包含的一些基因诱发了白血病，如由原癌基因的转录因子的不适当表达和异常表达造成。于是可以通过比对一些检验者的 DNA 序列数据和某些特殊的序列数据，之后通过比对结果就可以从检验者 DNA 序列数据中找出具有生物意义的特征，从而提供一些有价值的信息，这些信息为诊断白血病提供了一定的依据，从而对白血病的治疗方面提供一些指导性的作用。

1.2.3　国内外研究现状

多序列比对(multiple sequence alignment，MSA)问题是生物信息学中一个尚未解决的问题，被列为一个 NP 完全的组合优化问题，想要找到复杂性为多项式的精确算法是不可能的。多序列比对是生物序列分析的主要方法，也是生物信息学中挑战性的问题之一，Chuong B, Do 和 Katoh K(2008)在文中引用 258 个文献对多序列比对做了非常全面的综述，根据多序列比对的研究现状得出结论：虽然多序列比对问题已经研究了几十年，但是多序列比对的研究一直蓬勃发展，每一年国内外都有数十个描述多序列比对新方法的文章发表[如 McClure M A, et al.(1994), JD Notredame C(2002), Julie D. Thompson(2005), Chuong B Do, et al.(2008), C Kemena, et al.(2009), Orobitg Miquel, et al.(2013), 张鹏帅&霍红卫(2005), 葛宏伟&梁艳春(2006), 杨凡&朱清新等(2010)]。许多最近的研究表明，在提高比对的精度和比对处理范围可扩展性方面都取得相当大的进展，且在该方面仍有很大的发展空间。

1.2.4　多序列比对面临的挑战

同源性分析中常常要通过多序列比对来找出序列之间的相互关系。和 BLAST 的局部匹配搜索不同，多序列比对大多采用全局比对的算法。这样对于采用计算机程序的自动多序列比对是一个非常复杂且耗时的过程，特别是序列数目多且序列长的情况下。

多序列比对是一个 NP 完全问题，解决此问题的传统算法是渐进算法或迭代算法，但是随着序列长度和条数的增多，时空复杂性急剧上升，设计一个具有高敏感性、高精度且低复杂度的算法，成为解决生物多序列比对的瓶颈问题。

1.3　多序列比对的基本原理

1.3.1　多序列比对的相关概念

数据库搜索的基础是序列的相似性比对，而寻找同源序列则是数据库搜索的主要目的之一。在序列比对中需要注意区分以下几个基本概念。

1. 相似性、同源性

相似性(similarity)是指序列比对过程中用来描述检测序列和目标序列之间相同 DNA 碱基或氨基酸残基顺序所占比例的高低。例如，A 序列和 B 序列的相似性是 80%，这是个量化的关系。

同源性(homology)是指从一些数据中推断出的两个基因或蛋白质序列具有共同祖先的结论。例如，A 序列和 B 序列只有是同源序列或非同源序列两种关系，属于质的判断。

相似性和同源性是两个完全不同的概念。序列之间的相似程度是可以量化的参数，而序列是否同源则需要有进化事实的验证。任何序列之间均存在相似，只有当序列是从一个共同祖先进化分歧而来的，它们才是同源的。因此，相似的序列有可能同源，同源的序列也常常具有相似的生物学功能，但基因复制机制又使得同源序列进化出不同的功能。一般来说，序列间的相似性越高，它们是同源序列的可能性就越大，当相似度高于 50%时，比较容易推测检测序列和目标序列可能是同源序列；而当相似度低于 20%时，就难以确定或者根本无法确定其是否具有同源性，所以常常通过序列的相似性来推测序列是否同源。

(1) 序列相似性比较：将待测序列与 DNA 或蛋白质序列库进行比较，用于确定该序列的生物属性，也就是找出与此序列相似的已知序列是什么。完成这一工作只需要使用两两序列比较算法，常用的程序包括 BLAST、FASTA 等。

(2) 序列同源性分析：将待测序列加入到一组与之同源但来自不同物种的序列中进行多序列同时比较，以确定该序列与其他序列间的同源性大小。这是理论分析方法中最关键的一步。完成这一工作必须使用多序列比对算法，常用的程序包括 Clustal、MAFFT 等。

2. 直系同源、旁系同源

在考虑序列相似性时，还必须认识另外两个概念：直系同源(ortholog)和旁系同源(paralog)。分子进化不仅使不同的物种发生差异，在某个物种的基因组内部也可能发生进化事件。来自共同祖先的基因成为同源基因。

直系同源基因是指在不同物种中有相同功能的同源基因，它是在物种形成过程中形成的。旁系同源基因是指一个物种内的同源基因。一般情况下，一个生物物种的基因组中，两个基因或可读框在各自全长的 60%以上范围内，同一性不少于 30%时，称为同源基因。研究直系同源基因之间或旁系同源基因之间的功能关系，可以为基因组分析提供很大的帮助。直系同源基因由共同的祖先演化而来，从而具有序列相似性。而旁系同源是种内基因倍增的结果。当序列相似性高时，直系同源可以暗示功能性同源，而旁系同源一般会有相似但并不相同的功能。

1.3.2　序列比对的分类

1. 按比对的序列数量可分为双序列比对和多序列比对

双序列比对就是对两条序列进行比对，通过比较两个序列之间的相似区域和保守性位点，寻找二者可能的分子进化关系。双序列比对是序列分析的基础。然而，对于构成基因家族的成组的序列来说，要建立多个序列之间的关系，这样才能揭示整个基因家族的特征。

多序列比对就是对三条及以上的序列进行比对。多序列比对可用来区分一组序列之间的差异，但主要是用来描述一组序列之间的相似

性关系，以便对一个基因家族的特征有一个简明扼要的了解，在阐明一组相关序列的重要生物学模式方面起着相当重要的作用。

2. 按比对的序列长度可分为短序列比对和长序列比对

序列长度单位通常定义为 bp，一般规定长度不超过 100bp 的序列为短序列，超过 100bp 的序列为长序列。不同类型的测序数据适用于特定的实验应用领域：近似 100bp 长的双末端序列适合重测序研究；染色质免疫沉淀测序用于研究转录因子结合位点和组蛋白修饰位点，其序列长度不超过 50～75bp；对于基因组序列未知的物种测序，短序列组装通常是最基本的分析步骤，通过将短序列定位到参照基因组序列上进行后续分析。长序列比对对于特定基因组学问题来讲仍然是非常有意义的，例如基因组较大的物种基因组序列从头组装、重测序和结构变异检测等，目前也已经有很多小组基于 PacBio 的测序数据开展了一些相关研究并取得了非常好的效果。

3. 按比对的序列范围可分为全局比对和局部比对

全局比对考虑序列的全局相似性，是对给定序列全长进行比较的方式。运用全局比对的主要优势是对具有高度同源性的序列进行优化，这在以已知三维结构的同源性序列为基础对未知序列的三维结构进行预测的模型构建过程中是很有用的。

局部比对考虑序列片段之间的相似性，仅能获得特定序列在数据库中配对好的亚区。局部比对适合于那些在全长中具有局部的小同源性片段的序列比较，一般用于特定序列位点、结构域及其他类型重复序列的搜索，同时在发现数据库中待分析序列的同源序列过程中也具有重要意义。

1.3.3　多序列比对的数学定义

一条长度为 T 的序列是 T 个字符组成的字符串，字符取自于字

母表$\sum=\{A,G,C,T\}$，分别代表DNA的四个核苷酸类型。对于DNA序列，给定包含 N 个序列的序列集$S=\{S_1,S_2,\cdots,S_N\}$，$N\geqslant 2$，$S_i=S_{i1}S_{i2}\cdots S_{il_i}(1\leqslant i\leqslant N)$，$S_{ij}\in\sum(1\leqslant j\leqslant l_i)$，$l_i$是第$i$条序列的长度，则一个序列比对可定义为一个矩阵$\boldsymbol{A}=(a_{ij})$，其中$1\leqslant i\leqslant N, 1\leqslant j\leqslant l$，$\max(l_i)\leqslant l\leqslant\sum_{i=1}^{N}l_i$，如图1.1所示。矩阵必须满足下列三个条件：

(1) $a_{ij}\in\sum\cup\{-\}$，其中“－”代表空位。

(2) 矩阵中的第i行去掉“－”后，即得到字符串S_i。

(3) 矩阵中不包含字符全是空格的列。

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
I	Y	Y	D	G	G	A	V	—	E	A	L	C	A	M
II	Y	D	D	—	G	A	L	V	E	A	L	C	A	M
III	F	D	E	G	G	—	L	V	Q	A	—	C	F	—
IV	F	Y	E	G	G	I	V	V	Q	A	V	—	—	M
V	Y	D	D	G	G	I	L	V	—	—	L	C	A	N
VI	Y	Y	E	—	G	A	—	V	Q	A	V	C	E	M

图1.1　多序列比对

1.3.4　多序列比对的打分方法

序列分析的目的是揭示核苷酸或氨基酸序列编码的高级结构或功能信息，而序列比对打分的目的则是提供一个比较两(多)序列之间相似性的量度，从而使人们有可能迅速区分有着微妙不同的任意两个序列的比对结果。常见的打分标准有以下几种。

1. 目标函数

目标函数是用来考查多序列比对结果好坏的一种度量标准，所有的多序列比对方法都依赖于一个目标函数来说明比对结果的好坏，从而反映出此方法的精确度和有效性。当前有三种主流的目标函数：比对和函数(sum-of-pairs functions)。一致性函数(consensus

functions)和树函数(tree functions)，其中使用最普遍的是比对和函数(简称为 SP 函数)。SP 函数需要设置两个重要的参数：替换矩阵(substitution matrix)和空位罚分(gap penalties，其中包括起始空位罚分和延续空位罚分)。

sum-of-pair(SP) 函数的计算公式均可用 $score = \sum residue - \sum penalty$ 表示。其中，*score* 定义为正数，分值越高，比对效果越好；$\sum residue$ 是比对后的蛋白质序列中氨基酸残基的总分，定义为 $\sum residue > 0$；$\sum penalty$ 是插入空格产生的总罚分，定义为 $\sum penalty > 0$。

氨基酸残基的总分公式为

$$\sum residue = \sum_{h=1}^{L} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} cost(S_{ih}, S_{jh})$$

式中：

$$cost(S_{ih}, S_{jh}) = \begin{cases} S_{aa} = score(a,a) & \text{如果两个非空位残基相同(匹配)；} \\ S_{ab} = score(a,b) & \text{如果两个非空位残基不同(不匹配)；} \\ S_{a-} = score(a,-) = 0 & \text{残基与空位的分数为0(空位罚分另计)。} \end{cases}$$

式中：L 为比对序列长度；m 为参加比对的序列条数；S_{ih} 为第 i 条序列第 h 个残基。匹配和不匹配的分数通常由替换计分矩阵给出。

2. 替换计分矩阵

对于蛋白质序列，计分矩阵主要用于记录在做序列比对时两个相对应的残基的相似度。一旦这个矩阵定义好了以后，比对程式就可以利用这个矩阵，尽量将相似的残基排在一起，以达到最好的比对。常用的替换计分矩阵有可接受点突变矩阵(point accepted mutation，PAM)和模块替换矩阵(blocks substitution matrix，BLOSUM)。

1) PAM 矩阵

基于进化的点突变模型，如果两种氨基酸替换频繁，说明自然界接受这种替换，那么这对氨基酸替换得分就高。一个 PAM 就是一

个进化的变异单位，即 1%的氨基酸改变，但这并不意味 100 次 PAM 后，每个氨基酸都发生变化，因为其中一些位置可能会经过多次突变，甚至可能会变回到原来的氨基酸。对应于一个更大进化距离间隔的突变概率矩阵，可以通过对初始矩阵进行适当的数学处理得到(Dayhoff 等，1978)，如常用的 PAM250 矩阵。PAM250 相似性分数矩阵相当于在两个序列之间具有 20%的残基匹配，如图 1.2 所示。主对角线上分数值是指两个相同残基之间的相似性分数值，有些残基的分值较高，如色氨酸 W 为 17，半胱氨酸 C 为 12，说明它们比较保守，不易突变；有的残基的分值较低，如丝氨酸 S、丙氨酸 A、门冬酰胺 N 三种氨基酸均为 2，这些氨基酸则比较容易突变。不同氨基酸之间的分数值越高，它们之间的相似性越高，进化过程中容易发生互相突变，如苯丙氨酸 F 和酪氨酸 Y，它们之间的相似性分数值是 7；而相似性分数值为负数的氨基酸之间的相似性则较低，如甘氨酸 G 和色氨酸 W 之间为−7，它们在进化过程中不易发生互相突变。此外，图 1.2 所示矩阵中把理化性质相似的氨基酸按组排列在

		C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	Cys	12																			
S	Ser	0	2																		
T	Thr	-2	1	3																	
P	Pro	-3	1	0	6																
A	Ala	-2	1	1	1	2															
G	Gly	-3	1	0	-1	1	5														
N	Asn	-4	1	0	-1	0	0	2													
D	Asp	-5	0	0	-1	0	1	2	4												
E	Glu	-5	0	0	-1	0	0	1	3	4											
Q	Gln	-5	-1	-1	0	0	-1	1	2	2	4										
H	His	-3	-1	-1	0	-1	-2	2	1	1	3	6									
R	Arg	-4	0	-1	0	-2	-3	0	-1	-1	1	2	6								
K	Lys	-5	0	0	-1	-1	-2	1	0	0	1	0	3	5							
M	Met	-5	-2	-1	-2	-1	-3	-2	-3	-2	-1	-2	0	0	6						
I	Ile	-2	-1	0	-2	-1	-3	-2	-2	-2	-2	-2	-2	-2	2	5					
L	Leu	-6	-3	-2	-3	-2	-4	-3	-4	-3	-2	-2	-3	-3	4	2	6				
V	Val	-2	-1	0	-1	0	-1	-2	-2	-2	-2	-2	-2	-2	2	4	2	4			
F	Phe	-4	-3	-3	-5	-5	-5	-4	-6	-5	-5	-2	-4	-5	0	1	2	-1	9		
Y	Tyr	0	-3	-3	-5	-3	-5	-2	4	-4	-4	0	-4	-4	-2	-1	-1	-2	7	10	
W	Trp	-8	-2	-5	-6	-6	-7	-4	-7	-7	-5	-3	2	-3	-4	-5	-2	-6	0	0	17

图 1.2　PAM250 相似性分数矩阵

一起，如碱性氨基酸组氨酸 H、精氨酸 R 和赖氨酸 K。突变数据矩阵的产生基于相似性较高(通常为 85%以上)的序列比对，那些进化距离较远的矩阵(如 PAM250)是从初始模型中推算出来而不是直接计算得到的，其准确率受到一定限制。序列分析的关键是检测进化距离较远的序列之间是否具有同源性，因此突变数据矩阵在实际使用时存在一定的局限性。

针对不同的进化距离采用 PAM 矩阵：

序列相似度	=	40%	50%	60%
		|	|	|
打分矩阵	=	PAM120	PAM80	PAM60

PAM250→14%～27%

2) BLOSUM 矩阵

模块替换矩阵(BLOSUM)以序列片段为基础，基于蛋白质模块数据库 BLOCKS。Henikoff 夫妇(Henikoff 和 Henikoff，1992)从蛋白质模块数据库 BLOCKS 中找出一组替换矩阵，用于解决序列的远距离相关。在构建矩阵过程中，通过设置最小相同残基数百分比将序列片段整合在一起，以避免由于同一个残基对被重复计数而引入的任何潜在的偏差。在每一片段中，计算出每个残基位置的平均贡献，使得整个片段可以有效地被看作单一序列。通过设置不同的百分比，产生了不同矩阵。由此，高于或等于 80%相同的序列组成的串可用于产生 BLOSUM80 矩阵；那些有 62%或以上相同的串用于产生 BLOSUM62 矩阵，依此类推。BLOSUM 与 BLOCKS 对于同样的序列比对产生的结果在局部有所不同，可能是一个认为不相似不可以替换，而另一个认为相似可以替换。必须说明，如果比对这两个序列高度相似，这些细微的差别对整个序列比对结果的影响不大，但在序列比对的边界区可能产生显著影响，此时增强微弱信号以探测远距离相关变得十分重要。

此矩阵与 PAM 矩阵的不同之处在于：

(1) 用于产生矩阵的蛋白质家族及多肽链数目，BLOSUM 比

PAM 大约多 20 倍。

(2) PAM：家族内成员相比，然后把所有家族中对某种氨基酸的比较结果加和在一起，产生“取代”数据(PAM-1)，PAM-1 自乘 *n* 次，得 PAM-*n*；BLOSUM：首先寻找氨基酸模式，即有意义的一段氨基酸片断(如一个结构域及其相邻的两小段氨基酸序列)，分别比较相同的氨基酸模式之间氨基酸的保守性(某种氨基酸对另一种氨基酸的取代数据)，然后，以所有 60%保守性的氨基酸模式之间的比较数据为根据，产生 BLOSUM-60；以所有 80%保守性的氨基酸模式之间的比较数据为根据，产生 BLOSUM-80。

(3) PAM-*n* 中，*n* 越小，表示氨基酸变异的可能性越小。相似的序列之间比较应该选用 *n* 值小的矩阵，不太相似的序列之间比较应该选用 *n* 值大的矩阵。PAM-250 用于约 20%相同序列之间的比较。BLOSUM-*n* 中，*n* 越小，表示氨基酸相似的可能性越小。相似的序列之间比较应该选用 *n* 值大的矩阵，不太相似的序列之间比较应该选用 *n* 值小的矩阵。BLOSUM-62 用来比较 62%相似度的序列，BLOSUM-80 用来比较 80%左右的序列。

BLOSUM-62 矩阵如图 1.3 所示。

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	**9**	-1	-1	-3	0	-3	-3	-3	-4	-3	-3	-3	-3	-1	-1	-1	-1	-2	-2	-2
S	-1	**4**	1	-1	1	0	1	0	0	0	-1	-1	0	-1	-2	-2	-2	-2	-2	-3
T	-1	1	**4**	1	-1	1	0	1	0	0	0	-1	0	-1	-2	-2	-2	-2	-2	-3
P	-3	-1	1	**7**	-1	-2	-1	-1	-1	-1	-2	-2	-1	-2	-3	-3	-2	-4	-3	-4
A	0	1	-1	-1	**4**	0	-1	-2	-1	-1	-2	-1	-1	-1	-1	-1	-2	-2	-2	-3
G	-3	0	1	-2	0	**6**	-2	-1	-2	-2	-2	-2	-2	-3	-4	-4	0	-3	-3	-2
N	-3	1	0	-2	-2	0	**6**	1	0	0	-1	0	0	-2	-3	-3	-3	-3	-2	-4
D	-3	0	1	-1	-2	-1	1	**6**	2	0	-1	-2	-1	-3	-3	-4	-3	-3	-3	-4
E	-4	0	0	-1	-1	-2	0	2	**5**	2	0	0	1	-2	-3	-3	-3	-3	-2	-3
Q	-3	0	0	-1	-1	-2	0	0	2	**5**	0	1	1	0	-3	-2	-2	-3	-1	-2
H	-3	-1	0	-2	-2	-2	1	1	0	0	**8**	0	-1	-2	-3	-3	-2	-1	2	-2
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	**5**	2	-1	-3	-2	-3	-3	-2	-3
K	-3	0	0	-1	-1	-2	0	-1	1	1	-1	2	**5**	-1	-3	-2	-3	-3	-2	-3
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	**5**	1	2	-2	0	-1	-1
I	-1	-2	-2	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	**4**	2	1	0	-1	-3
L	-1	-2	-2	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	**4**	3	0	-1	-2
V	-1	-2	-2	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	**4**	-1	-1	-3
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	**6**	3	1
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	-2	-2	-2	-1	-1	-1	-1	3	**7**	2
W	-2	-3	-3	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	**11**

图 1.3　BLOSUM-62 计分替换矩阵

应用 BLOSUM-62 矩阵(为示例表述清晰，只截取部分矩阵内容)计算双序列比对残基分数的示例见图 1.4。

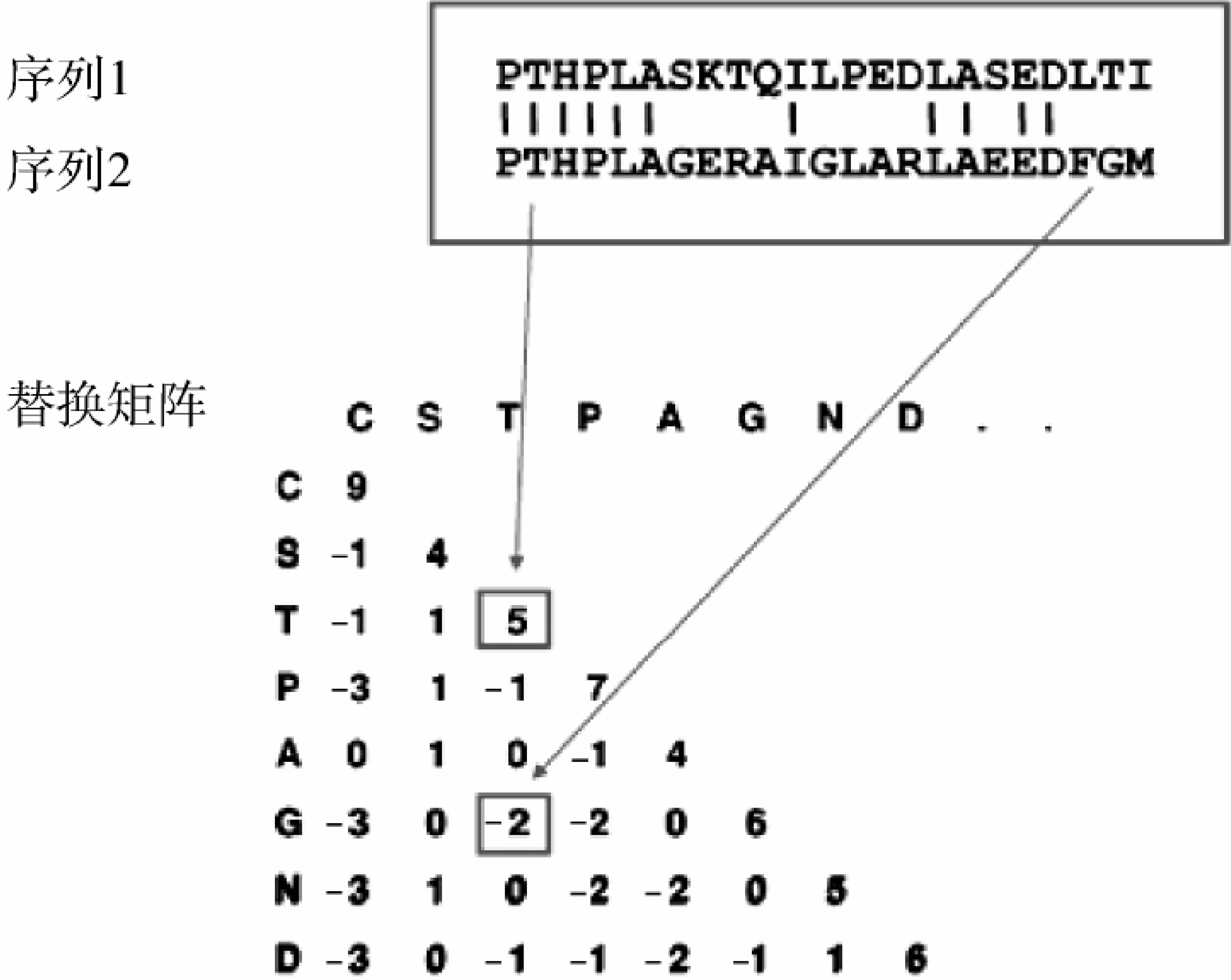


图 1.4　残基分数计算图

在这个例子中，共有两条序列，即 m=2，每个序列有 23 个残基，且长度相等，即 L=23，对照图 1.3 所示矩阵，有 $S_{\mathrm{TT}}=5$，$S_{\mathrm{TG}}=-2$ 等比对数据，则

$$\sum residue=\sum_{h=1}^{L}\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}cost(S_{ih},S_{jh})=\sum_{h=1}^{23}\sum_{i=1}^{1}\sum_{j=2}^{2}cost(S_{ih},S_{jh})=48$$

3. 空位罚分

在比对过程中需要在检测序列或目标序列中引入空位(gap)，以表示插入和删除。一般情况下，参与比对的多条序列不完全相同，为了将其对齐，就需要插入空位。为了使整条序列而不仅仅是空位插入区域产生可比较的模式，空位的插入是必需的。不插入空位，序列比对过程就无法进行。序列对齐后，才能提出最初的同源性假设，才能确定插入缺失置换(transition)和颠换(transversion)等事件是

否发生以及发生位置和发生频率。

在多序列比对阶段，如果选择插入大量的空位，那么任意两条随机的不相关的序列都可以对齐，但是这样的比对结果可能毫无生物学意义，因此必须在一定程度上限制空位的数目以产生有生物学意义的比对结果，为此采用一个打分策略：匹配的残基获得正的分值，而空位获得负的分值或者叫罚分(Hall，2001)，这样获得最大净分值的比对结果即为比对程序所寻找的最优结果。

空位罚分包括两部分：起始空位罚分和延伸空位罚分。一般起始空位罚分高于延伸空位罚分。所谓起始空位，是指序列比对时在某一序列中插入的第一个空位。所谓延伸空位，是指在引入一个或几个空位后，继续引入下一个连续的空位。空位罚分$\sum penalty$有以下两种计算方法。

1) 线性空位罚分

线性空位罚分(constant gap penalty)是最简单的罚分方式，对于每一个空位罚同样的分数，则总空位罚分的计算公式为$\sum penalty = n \times gap$。其中，$n$是空位的个数；$gap$是空位的罚分。

对于图 1.5 示例，假设空位罚分为 5，则该例的总空位罚分是 11×5=55 分。

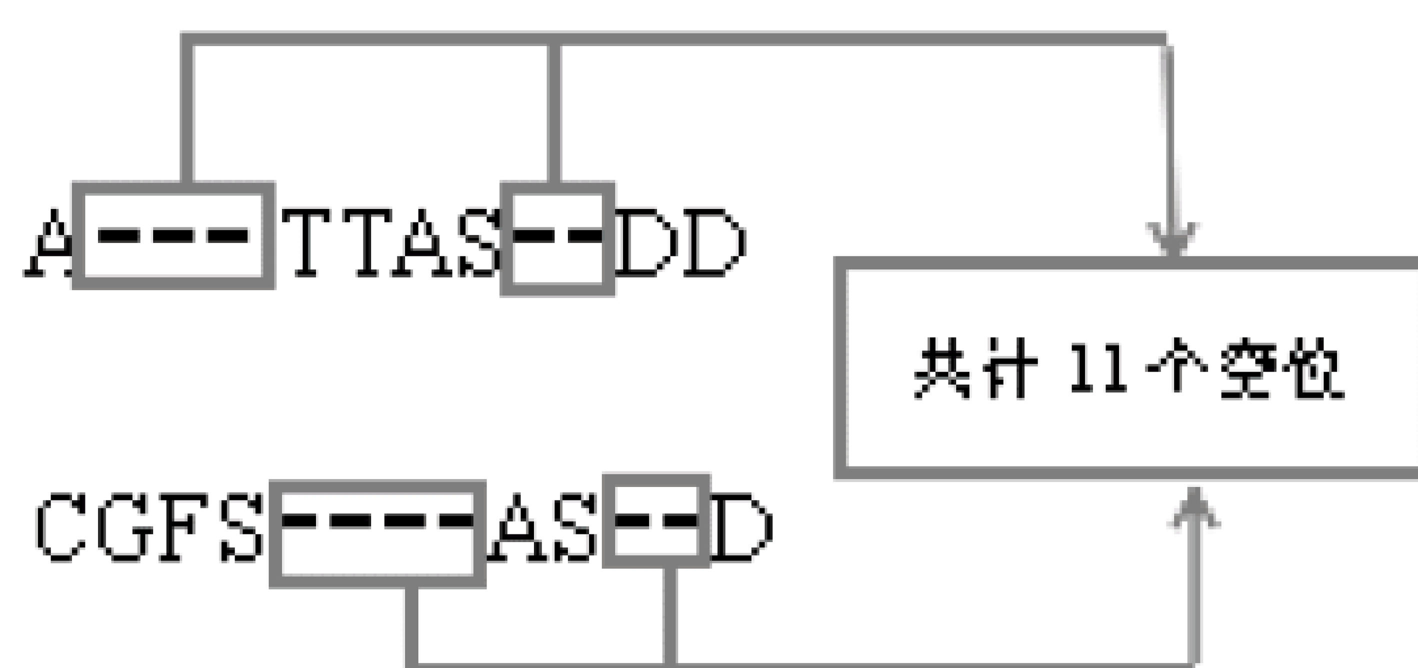


图 1.5　线性空位罚分计算

2) 仿射空位罚分

大多数的比对算法采用仿射空位罚分策略，这种策略设定起始空位罚分加权起始空位，并以一个较小的延伸空位罚分来加权延伸

空位，起始空位罚分(gap open penalty)和延伸空位罚分(gap extend penalty)的值域分别为 0～100 和 0～10。研究表明，增加起始空位罚分会使插入空位频率降低，增加延伸空位罚分则使空位变短。起始空位后的延伸空位被赋予较小的罚分值，用以鼓励长的且更加连续的空位而不是大量单个空位的插入。

起始空位罚分和延伸空位罚分的分值直接影响到序列比对的结果，如表 1.1 所示。

表 1.1　起始空位罚分和延伸空位罚分对比对的影响

起始空位罚分	延伸空位罚分	说　明
大	大	插入和删除极少，适用于相似性较高的序列比对
大	小	少量的大片空位插入
小	大	大量小块插入，用于相似性较低的序列比对

仿射空位罚分根据生物定义将空位分为起始空位(gap open penalty)和延续空位(gap extend penalty)，其产生的罚分分别简写为 GOP 和 GEP，$\sum penalty = N_{GOP} \cdot GOP + N_{GEP} \cdot GEP$。其中，$N_{GOP}$ 是 GOP 的个数，N_{GEP} 是 GEP 的个数，且 GOP>GEP。具有生物意义的仿射罚分是当前最常应用的罚分方式。图 1.5 示例以仿射空位罚分计算，则如图 1.6 所示。

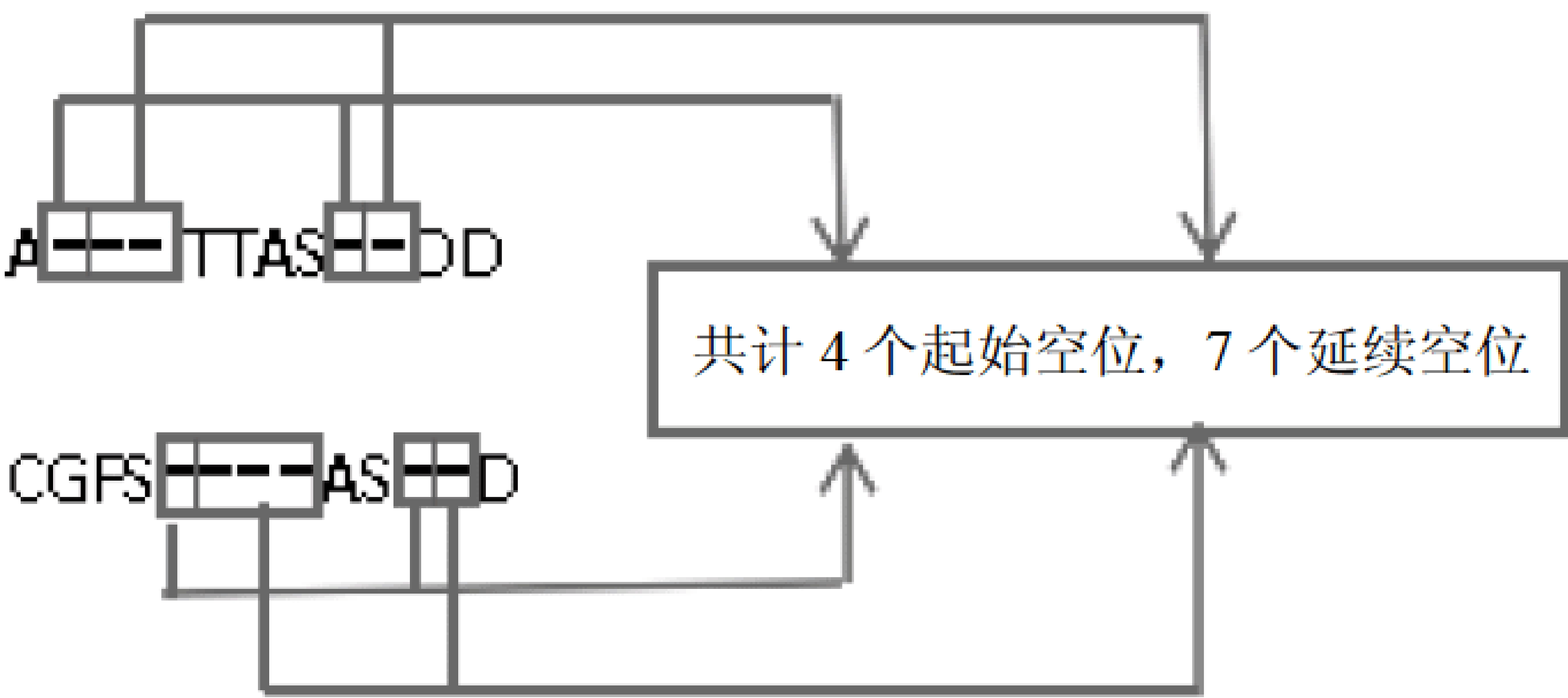


图 1.6　仿射空位罚分计算

假设起始空位罚分为 5，延续空位罚分为 1，则该例的总空位罚分是 4×5+7×1=27 分。

1.4　多序列比对方法

1.4.1　比对方法

1. 手工比对

手工比对方法就是通过辅助编辑软件(如 BioEdit、SeaView、GeneDoc 等软件)的不同颜色显示不同残基，靠分析者的观察，手工改变比对的状态。手工比对方法在文献中经常看到。因为在手工比对过程中难免会有一些主观因素，通常被认为有较大的随意性。

通常使用不同颜色表示具有不同特性的残基，以便判别序列间的相似性。颜色的选择非常重要，如果使用不当，则看起来不够直观，就会丢失一些比对结果中的有用信息。反之，如果选择恰当，就能从比对结果中快速找到某些重要的结构模式和功能位点。颜色的选择可以根据主观愿望和喜好，但是最好和常规方法一致。表 1.2 给出了氨基酸分组方法和代表性颜色。

表 1.2　氨基酸分组方法和代表性颜色

残基种类	残基特性	颜色
Asp(D)，Glu(E)	酸性	红色
His(H)，Arg(R)，Lys(K)	碱性	蓝色
Ser(S)，Thr(T)，Asn(N)，Gln(Q)	极性	绿色
Ala(A)，Val(V)，Leu(L)，Ile(I)，Met(M)	疏水性，带支链	白色
Phe(F)，Tyr(Y)，Trp(W)	疏水性，带苯环	紫色
Pro(P)，Gly(G)	侧链结构特殊	棕色
Cys(C)	能形成二硫键	黄色

2. 计算机程序自动比对

通过特定的算法(如同步法、渐进法等算法，详见 1.4.2 节)，由计算机程序自动搜索最佳的多序列比对状态。

(1) 同步法：将序列两两比对时的二维动态规划矩阵扩展到三维矩阵；即用矩阵的维数来反映比对的序列数目。这种方法的计算量很大，对于计算机系统的资源要求较高，一般只有在进行少数的较短的序列比对时才会用到这种方法。

(2) 渐进法：最常见的是 Clustal 所采用的方法，其基本思想就是基于相似序列通常具有进化相关性的假设。

1.4.2　多序列比对算法

在生物多序列比对方面，现有的算法基本分为三大类：精确比对算法、近似比对算法和基于图论的比对算法。

1. 精确比对算法

精确比对算法是基于数学理论基础上的一种动态规划算法。利用动态规划思想求解序列比对问题的方法最早由 Needleman 和 Wunsch 于 1970 年联合提出，并将该算法用于求解两条蛋白质序列的全局比对问题，因此该方法也称为 Needleman-Wunsch 算法，被视为经典的双序列比对全局动态规划算法。1981 年，Smith 和 Waterman 在改进 Needleman-Wunsch 算法的基础上提出了经典的双序列比对局部动态规划算法——Smith-Waterman 算法，该算法适用于亲缘关系较远、整体上不具有相似性但在一些较小的区域上存在局部相似性的两条序列。动态规划算法的思想核心是将原问题分解为子问题，基本步骤如下：

(1) 最优分的递归计算。

(2) 存储子问题的最优分的动态规划矩阵。

(3) 子问题最优解矩阵的填充过程。

(4) 寻找最优比对路径的回溯方法。

精确比对算法的优点是比对非常精确，不会重复算或漏算。通过此方法虽然能得到理论上的最佳值，可是现实中并不常采用此方法，因为此算法的空间和时间的复杂度都会随着序列条数和序列的长度成指数级的速度增长。由于动态规划消耗的时间开销太大，因此一般只用于双序列比对问题的求解中。动态规划算法是生物信息学中一个最流行的解决方法，序列的比较、基因的识别、蛋白质序列的重排以及蛋白质结构和功能的分析等很多生物信息学中的问题都可以通过动态规划的方法解决。但是基本动态规划方法的时间和空间复杂度为$O(m \times n)$，满足不了实际的需要，所以，后人在此基础上提出了各种各样的基于动态规划思想的改进算法。下面介绍经典的两类双序列动态规划算法。

1) Needleman-Wunsch 算法

Needleman-Wunsch 算法是用在蛋白质序列的双比对问题上的经典全局动态规划算法，该算法从整体水平上分析不同的两条序列之间的进化关系或者同源关系，即考虑序列总长度的比较，用类似于使整体相似最大化的方式，对序列进行比对。如果参与比对的是长度互不相同的几条序列，则需要采用一些方法在序列的某些位置插入空格，从而使序列的长度达到一致。Needleman-Wunsch 算法的基本思想是：使用递归方法计算出两条序列所有可能的比对结果的相似性得分，同时将该得分存储在某个矩阵中，该矩阵称为得分矩阵。如果将矩阵中每一个分值所在单元格的位置称为一个单元，则从一个单元移动到另一个单元的路径或者说方向最多有三种：向上、向左和向左上。回溯时从右下角开始，每次都选择相邻的最大元素，并用箭头对所选的路径做出标记，直到到达矩阵的左上角时才算结束。这时根据标出的路径，通过动态规划的方法回溯，则能寻找出最优的相似性比对。

设序列S和T的长度分别为m和n。考虑S和T的两个前缀$s[0,i]$和$t[0,j]$，$i,j>1$。假设已知序列$s[0,i]$和$t[0,j]$所有较短的连续子

序列的最优比对，即已知：

(1) $s[0,i-1]$和$t[0,j-1]$的最优比对；

(2) $s[0,i-1]$和$t[0,j]$的最优比对；

(3) $s[0,i]$和$t[0,j-1]$的最优比对。

则$s[0,i]$和$t[0,j]$的最优比对一定是上述三种情况之一的扩展，即：

(1) 替换(s_i,t_j)或匹配(s_i,t_j)，这取决于s_i是否等于t_j；

(2) 删除$(s_i,-)$；

(3) 插入$(-,t_j)$。

令$M(s[0,i],t[0,j])$为序列$s[0,i]$和$t[0,j]$对比的得分，可根据式(1.1)所示的递归算式计算最大值：

$$M(s[0,i],t[0,j])=\max\begin{cases}M(s[0,i-1],t[0,j]+\sigma(s_i,-))\\M(s[0,i-1],t[0,j-1]+\sigma(s_i,t_j))\\M(s[0,i],t[0,j-1]+\sigma(-,t_j))\end{cases}\tag{1.1}$$

其初值为
$$\begin{cases}M(s[0,0],t[0,0])=0\\M(s[0,i],t[0,0])=M(s[0,i-1],t[0,0]+\sigma(s_i,-))\\M(s[0,0],t[0,j])=M(s[0,0],t[0,j-1]+\sigma(-,t_j))\end{cases}\tag{1.2}$$

按照这种方法，对于给定的打分函数$\sigma(s_i,t_j)$，两条序列所有前缀的比对得分值定义了一个$(m+1)\times(n+1)$的得分矩阵：

$$\boldsymbol{D}=(d_{i,j})\tag{1.3}$$

式中，$d_{i,j}=M(s[0,i],t[0,j])$。

对于一个长度为 n 的序列，有$n+1$个前缀(包括一个空序列)，所以得分矩阵的大小为$(m+1)\times(n+1)$。其中矩阵的纵轴方向自上而下对应于第一条序列 S，横轴方向从左到右对应于第二条序列 T。矩阵横向移动表示在纵轴序列中加入一个空位，纵向的移动表示在横轴序列中加入一个空位，而斜对角向的移动表示两序列各自相应的字符进行比对，各轴第一个元素的索引下标为 0。

$d_{i,j}$ 的计算公式为

$$d_{i,j}=\max\begin{cases}d_{i-1,j}+\sigma(s_i,-)\\d_{i-1,j-1}+\sigma(s_i,t_j)\\d_{i,j-1}+\sigma(-,t_j)\end{cases}\tag{1.4}$$

$d_{i,j}$ 最大值的三种选择决定了各矩阵元素之间的关系，如图 1.7 所示。

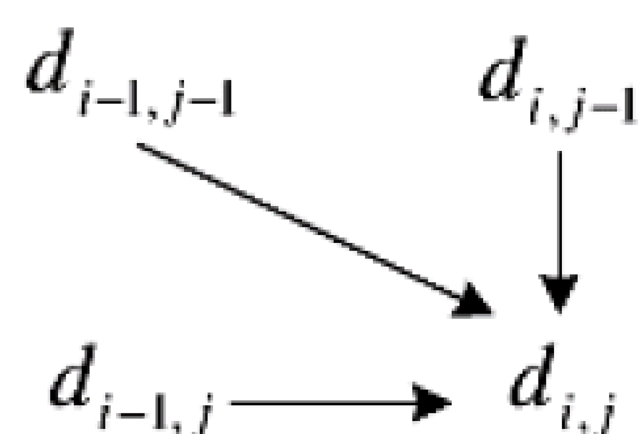


图 1.7　矩阵各元素的关系

矩阵右下角元素即为期望的结果：$d_{m,n}=M(s[0,m],t[0,n])=M(s,t)$。其计算过程描述如下：首先初始化得分矩阵 **D**，然后计算 **D** 矩阵的其他元素。计算过程从 $d_{0,0}$ 开始，可以按行计算，每行从左到右，也可以按列计算，每列从上到下。任何计算过程，只要满足在计算 $d_{i,j}$ 时 $d_{i-1,j}$、$d_{i-1,j-1}$、$d_{i,j-1}$ 都已经被计算这个条件即可。在计算 $d_{i,j}$ 后，需要保存 $d_{i,j}$ 是从 $d_{i-1,j}$ 或 $d_{i,j-1}$ 中的哪一个推进的，或保存计算的路径，以便于后续处理。上述计算过程到 $d_{m,n}$ 结束。

与计算过程相反，求最优路径或最优比对时，从 $d_{m,n}$ 开始，反向前推。假设在反推时到达 $d_{i,j}$，现在根据保存的计算路径判断 $d_{i,j}$ 究竟是根据 $d_{i-1,j}$、$d_{i-1,j-1}$、$d_{i,j-1}$ 中的哪一个计算而得到的。找到这个点以后，再从此点出发，一直到 $d_{0,0}$ 为止。走过的这条路径就是最优路径(即得分最大路径)，其对应于两条序列的最优比对。

根据算法原理可以计算出 Needleman-Wunsch 算法在双序列比对的时间复杂度为 $O(m\bullet n)$。三条序列比对可以理解为将双序列比对的二维空间扩展到三维空间，类似于数学中的三维坐标系，即在原来的二维平面上增加一条坐标轴，此时算法的时间复杂度就变成

了 $O(m \bullet n \bullet p)$， p 是第三条序列的长度。依此类推，当用 Needleman-Wunsch 算法解决 N 条序列比对的时间复杂度为 $O(l_1 \bullet l_2 \bullet l_3 \bullet \cdots \bullet l_N)$，$l_N$ 是第 N 条序列的长度。

2) Smith-Waterman 算法

在生物学中，残基的功能点是由较短的序列片段组成的，亲缘关系较远的两条蛋白质序列可能只存在一些相同的基因片段，因此寻找出具有最大相似度的片段对于物种亲缘关系的测定具有重要意义。1981 年，Smith 和 Waterman 在改进 Needleman-Wunsch 算法的基础上提出了 Smith-Waterman 算法。该算法是经典的双序列局部比对动态规划算法，适用于亲缘关系较远、整体上不具有相似性但在一些较小的区域上存在局部相似性的两条序列。

Smith-Waterman 算法的思想与 Needleman-Wunsch 算法基本相似，仍然采用动态规划思想，记分矩阵的方式在识别局部相似性时灵敏度非常高。只不过在 Smith-Waterman 的记分矩阵中元素是根据式(1.5)所得，且三个记分函数的值也有相应的变化。

$$M(s[0,i],t[0,j]) = \max \begin{cases} M(s[0,i-1],t[0,j] + \sigma(s_i,-)) \\ M(s[0,i-1],t[0,j-1] + \sigma(s_i,t_j)) \\ M(s[0,i],t[0,j-1] + \sigma(-,t_j)) \end{cases} \tag{1.5}$$

其初值为

$$\begin{cases} M(s[0,0],t[0,0]) = 0 \\ M(s[0,i],t[0,0]) = 0 \\ M(s[0,0],t[0,j]) = 0 \end{cases} \tag{1.6}$$

因此，Needleman-Wunsch 算法和 Smith-Waterman 算法的不同点在于前者的回溯从记分矩阵的右下角开始，回溯到左上角终止；后者的回溯从记分矩阵中的最大元素开始，终止于第一个遇到的 0 元素。

2. 近似比对算法

近似比对算法又称启发式算法。目前国际上最具代表性的启发式算法有两大类：渐进比对算法和迭代比对算法。

1) 渐进比对算法

渐进比对算法又称贪心算法，最开始是由Hogeweg提出的，Feng和 Doolittle 等改进了此算法，并开发出了现在经常使用的序列比对程序软件包 Clustal、ClustalW 和 T-Coffee。这种算法的思路是：首先将多个序列两两比对构建距离矩阵，反映序列之间的两两关系；然后根据距离矩阵计算系统进化指导树，对关系密切的序列进行加权；最后从关系最紧密的两条序列开始，逐步引进临近的序列并不断重新构建比对，直到所有序列都被加入为止。渐进比对算法由于简单快速的特点，使它成为多序列比对中最常用的方法之一，但由于渐进比对算法的本质为贪心算法，一次比对的部分结果不能随着比对过程中更多序列的加入而改变，因此由于渐进过程中部分比对结果是“冻结”的，从而导致“局部最小化”问题的产生。

ClustalW 是一个最常用、最经典的基于渐进比对算法的多序列比对程序。ClustalW 采用近邻法生成向导树，此算法对树的每一个分枝长度的估算更准确，这样依据分枝长度计算的序列权重也就更加准确。另外，ClustalW 所采用的空位罚分策略比较复杂，影响空位罚分的因素有残基特异性、序列长度、比对时采用的记分矩阵、序列相似程度、空位位置等，这就使得空位的产生更加合理，从而提高了比对的准确度。T-Coffee 也是一个采用渐进比对算法的多序列比对程序，它与 ClustalW 最大的不同在于：前者采用 SP 记分函数作为目标函数，而 T-Coffee 采用 Coffee 记分函数作为目标函数，这样可以有效减少比对初期的错误，使得 T-Coffee 可以快速、准确地进行序列比对。测试结果表明 T-Coffee 比对的准确度高于 ClustalW，但其比对速度低于 ClustalW。Iain M.Wallace 提出了一种 M-Coffee 的

方法，此方法是在 T-Coffee 的基础上进行的。Katoh K 提出的 MAFFT 是一个引入快速傅里叶变换的快速多序列比对程序，它将每一条序列转化为包含每一个残基的体积和极性值的序列，然后采用快速傅里叶变换算法来寻找序列间的同源模块，从而减小动态规划的矩阵空间。MAFFT 中的 FFT-NS_2 基于渐进比对算法实现，测试结果表明其比对速度快于 ClustalW。Edgar R C 提出的 MUSCLE 方法在时间复杂度和空间复杂度上进行了改进。

2) 迭代比对算法

迭代比对是另一类有效的应用广泛的多序列比对策略，与渐进比对算法不同，它基于一个能产生比对的算法，并通过一定的进化策略，不断迭代替换来精细多序列比对，直到比对结果不再改进为止。这类算法的缺点是不能提供获得优化比对结果的保证，速度也不能和渐进比对算法相比。优点是将目标函数和优化过程在概念上进行了分离，具有鲁棒性、对于序列比对个数不敏感等特性。

近年来，迭代算法也被越来越多地应用到序列比对中去，如遗传算法、蚁群算法、隐马尔科夫等。Cedric Notredame 首先提出用遗传算法解决多序列比对问题，在遗传算法中对 22 种交叉变异算子应用了一个自动调度机制，并证实了此算法的有效性，其准确度与 ClustalW 结果相似。Notredame 提出了将遗传算法应用在 RNA 序列比对上，并取得了很好的比对结果。Thomsen 研究发现，使用自动调度策略得到的结果并不比以同等概率选择遗传算子好，因此自动调度策略的使用并非改善其算法的准确性，反而使得算法的复杂度增高。Goondro C 认为初始化种群的质量直接影响到算法的收敛速度，适应度值高的种群能够很快地收敛到接近最优解的解，因此他提出了一种新的初始化种群的方法，以增高初始种群的适应度值。Fernando Jose Mateus da Silva 提出了将局部最优搜索融入遗传算法中的新算法，提高了算法的准确度。胡桂武提出了一种基于遗传算法与星比对算法的多序列比对混合算法，这种算法是先通过星比对

算法得到一个多序列比对，然后将这个比对作为种群中的一个个体，并结合遗传算法的一种混合算法。张维梅提出了一种基于遗传算法和蚁群算法的多重序列比对算法，这种算法是将蚁群算法作为局部搜索的一种算法。司秀华提出了一种多搜索策略的多生物序列比对自适应遗传算法，这种算法是通过调整遗传算法中的交叉率和变异率从而避免算法找到局部最优而提出的。司徒浩臻提出了基于遗传算法的多序列比对算法。还有许多的求解多序列比对的混合算法，如遗传算法和蚁群算法相结合的求解多序列比对的方法、遗传算法和 GLOCSA 相结合的序列比对方法、并行混合遗传算法，还有一些以遗传算法为主的序列比对算法。除了这些算法外，还有许多其他的方法，如 Taheri J 提出的两种求解多序列比对的方法 RBT-L 和 RBT-GA。Fan H 提出了智能算子在遗传算法中的应用。邹权和郭茂祖等综述了常见的启发式方法，除了遗传算法和粒子群算法，还有许多其他的启发式方法，尽管在应用到多序列比对问题上做了许多尝试，但中间还存在一些十分难处理的问题，因而还没有形成基于这些方法的主流软件。

根据迭代算法的特点，本书以迭代算法作为解决多序列比对的主要方法，包括遗传算法、粒子群算法、量子粒子群算法以及结合 HMM 的粒子群算法等方法，在后面将详细介绍这些迭代算法的基本原理与应用。

3. 基于图论的比对算法

应用图论解决多序列比对问题的思想是近年来提出并发展起来的一种新方法，它与动态规划算法、渐进比对算法和迭代比对算法有着根本上的不同。

这种算法的基本思路是通过将序列中所有的片段转化成一个 DeBruijn 图，将序列装配问题转变成欧拉路径问题，这种方法称为“欧拉比对”。利用图论方法进行多序列比对最重要的优势是在比对

过程中所需要的时间和空间与序列的长度成线性关系，并且可以提高比对精度。全局多序列比对问题实际上就是多个极端片段的装配问题，如图 1.8 所示，如果几乎所有的片段都来自于基因组的相同区域，欧拉比对方法就会输出该区域的一致性序列。对于多序列比对，如果一致性序列是与所有给定序列最接近的一个，那么就希望通过某种记分机制提高序列的一致性。欧拉比对将原始的复杂图转化成有向无环图(directed acyclic graph，DAG)，并且在转化过程中尽可能记住这些序列的 k 元组，而且 DAG 图的边的权重记录的是序列的一致性的最大值。

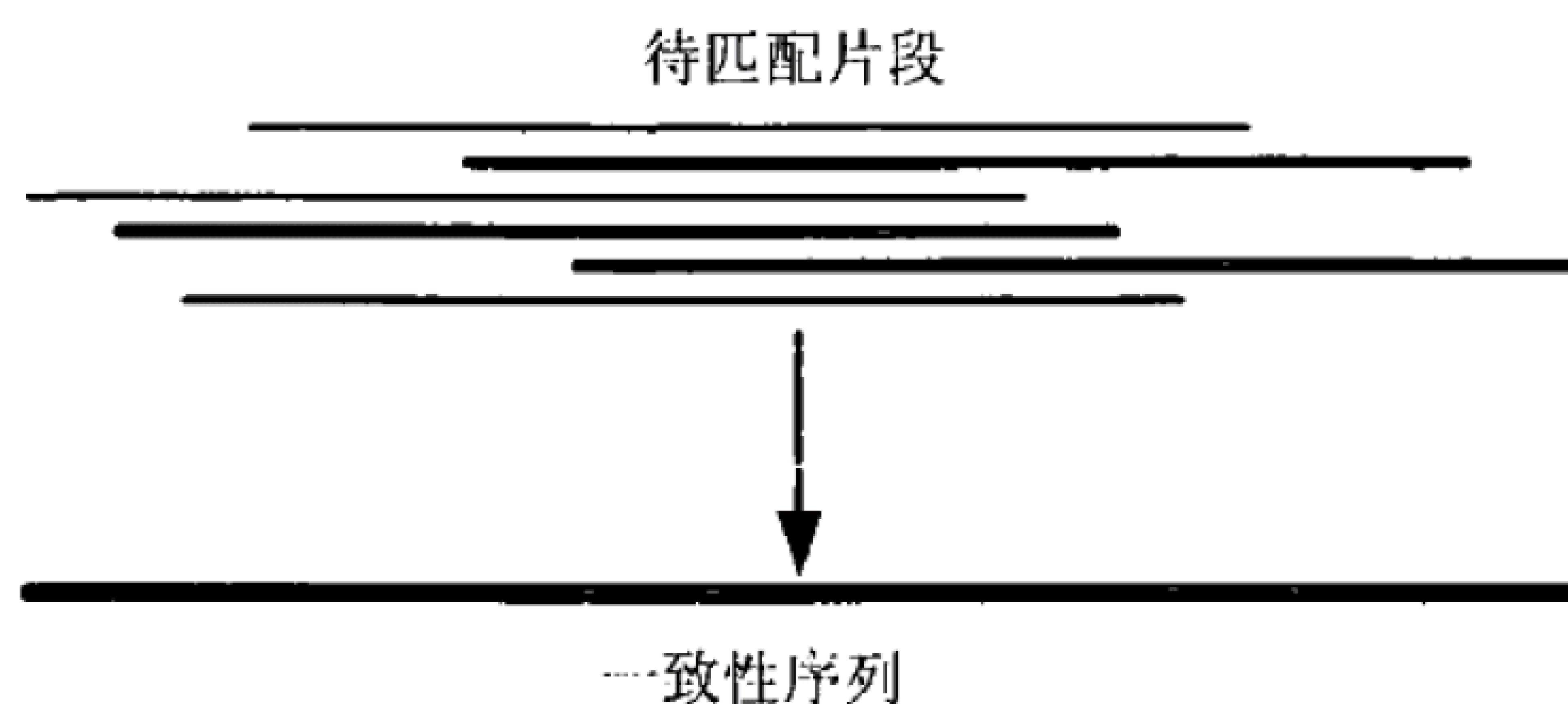


图 1.8　序列比对问题转化为序列装配问题

通过这两步，寻找一致性序列的问题就成了寻找最大路径的问题，寻找的主要流程是：①用序列中需要比对的 k 元组构造有向 DeBruijn 图；②将有向 DeBruijn 图转换成 DAG；③根据 DAG 各条边的权重求出一致性路径；④在一致性路径和每个输入序列之间作一次快速双序列比对；⑤根据双序列比对构造最后的多序列比对结果。

基于图论的多序列比对方法的主要代表就是 Lee 和 Grasso 等提出的偏序比对方法(partial order alignment，POA)。POA 能直接根据动态的双序列比对程序自动地进行比对，避免了将 MSA 降低维数的麻烦，从而保证每一个新序列与 MSA 中的序列的最优比对都能被考虑到。在 POA 算法中，新的编辑操作——同源重组在多

区域序列中起着重要作用，以有向无环图(DAG)的表示方式取代了过去 30 多年来用线性行列表示多序列比对的方式。在行列表示的比对方法中，图总是单一的有向路径，而 POA 方法扩展了图的结构，使得它成为一个有向无环图，打破了这个限制，在多序列比对领域中打开一个全新的视角。后来 Yuzhen Ye 和 Adam Godzik 把偏序结构又进一步应用于蛋白质序列比对中。Benjamin Raphael 等在 2004 年提出了 ABA(A-Bruijn alignment)算法。ABA 算法与以前的方法不同点在于：ABA 以有向图的方式表示一个比对，在这个图中允许环的存在。这种表示方式使得 ABA 算法比传统的比对矩阵甚至偏序比对(POA)更具有灵活性，尤其适用于含有交错或重复区域结构的蛋白质序列。允许构建包含下面三类区域的蛋白质序列：①不是在所有蛋白质序列中都出现的区域；②在不同的蛋白质中以不同顺序出现的区域；③在某些蛋白质序列中重复出现的区域。ABA 在求解包含重复和倒位的 DNA 序列比对问题时非常适用。霍红卫提出了用于进行全局 DNA 多序列比对的基于最大权值路径算法的 DNA 多序列比对方法。

现有的多序列比对程序大多基于上述算法思想或多种算法思想的结合并且选择不同的目标函数。表 1.3 列出了部分研究成果。

现有的多序列比对程序各有其优缺点，有文献对部分比对程序做了比较，所得结论为不存在唯一最好的比对程序，在进行序列比对时，根据问题特性，选用不同的序列比对程序，以期得到最满意的比对结果。而对于所得到的比对结果也不能简单地做出“正确或错误”的结论，因为多序列比对的方法建立在某个数学或生物学模型上，所以只能认为所使用的模型在多大程度上反映了序列之间的相似性关系以及它们的生物学特性。

表 1.3　多序列比对算法统计表

程序名称	算法
MSA	准确比对，基于 Carrillo-Lipman 算法
DCA	准确比对，基于分治法
CLUSTALW	渐进比对，基于动态规划方法，SP 记分函数
T-Coffee	渐进比对，基于动态规划方法，COFFEE 记分函数
Praline	渐进比对和迭代比对结合
IterAlign	迭代比对
Prrp	随机迭代比对
HMMT	随机迭代比对，基于 Markov 模型
SAGA	随机迭代比对，基于遗传算法，SP 和 COFFEE 记分函数可选
PHGA	随机迭代比对，基于并行遗传算法，SP 记分函数
MAFFT	其中 FFT-NS-2 采用渐进比对，FFT-NS-1 采用迭代比对
MUSCLE	渐进比对
Align-m	渐进比对
PROBCONS	渐进比对
POA	图论比对
ABA	图论比对

1.5　多序列比对常用数据库

随着生物数据的爆炸增长，目前记录的核苷酸序列已有成千上万条，蛋白质序列超过 10 万条。如此巨大数量的资源，必须应用电子数据库的存储和计算机分析。在对生物序列进行分析的过程中，除了由生物学实验直接得到实验数据信息外，还可以通过

查询相关的生物学数据库来得到相关的数据信息，特别对于数据分析方法的确立和研究，大量已知的数据信息是必须和非常重要的。虽然目前有很多不同类型的数据库，但这里将结合本书的主题，根据数据库的用途进行分类，分别介绍两种常用的生物序列数据库，通过应用数据库可以得到核酸序列和氨基酸序列的基本信息查询和序列比对算法的验证，为基因与蛋白质的深入分析提供可参考的信息。

1.5.1　综合性数据库

这类数据库中包含的生物数据信息非常齐全，通常应用于搜索查询相关的生物数据。以下是两个常用的数据库中心，它们提供了100 多种不同数据库的链接。

1. NCBI(美国国家生物技术信息中心)

NCBI(http://www.ncbi.nlm.nih.gov)全称为 National Center of Biotechnology Information。参议员 Claude Pepper 意识到信息计算机化过程方法对指导生物医学研究的重要性，发起了在 1988 年 11 月 4 日建立国立生物技术信息中心(NCBI)的立法。NCBI 是 NIH 的(美国)国立医学图书馆(NLM)的一个分支。NLM 是因为它在创立和维护生物信息学数据库方面的经验被选择的，而且这可以建立一个内部的关于计算分子生物学的研究计划。NCBI 的主要任务是发展新的信息学技术来帮助对那些控制健康和疾病的基本分子和遗传过程的理解。NCBI 有核酸、蛋白、基因名、基因组名等搜索工具，GenBank 数据库、BLAST 序列比对搜索工具，PUBMED 文献数据库，Taxonomy 数据，COG 蛋白家族库等。FTP 可以下载它全部的数据库、BLAST 的单机程序，以及各种工具程序。到目前为止，NCBI 已成为世界级的生物信息资源中心，为生物医学和生命科学研究提供了大量数据和分

析工具的平台。

2. EBI(欧洲生物信息研究所)

EBI(http://www.ebi.ac.uk/)全称是 European Bioinformatics Institute。EBI 拥有超过 20 年生物信息学研究和服务经验，是全球收集和传播生物数据、提供免费生物信息服务的欧洲节点。该所管理维护着世界最全面的分子生物数据库，其中很多是生物学家熟悉的数据库，如 ENA(核酸序列数据库)、Ensembl(基因组)、ArrayExpress(基因表达数据)、UniProtKB 蛋白质序列、InterPro(蛋白质家族/域/蛋白指纹等)和 PDBe(大分子结构)。ENA 在原 EMBL-Bank 核酸序列数据库基础上发展起来，是欧洲最重要的核酸序列资源，与美国 NCBI 的 GenBank 和日本的 DDBJ 组成国际核酸序列数据库合作联盟(INSDC)。这三大数据库各自收录了世界上所报道的所有序列数据的一部分，并且每天实时更新交换各自的序列信息。EBI 的数据资源包括 IntAct(蛋白质相互作用)、Reactome(反应途径)、ChEBI(小分子)等。EBI 向全球提供免费的生物信息服务，发展和维护着多种用于浏览、检索、分析处理生物数据的工具服务。数据获取工具 SRS(序列检索系统)为用户提供了快速、便捷和友好的界面以搜索超过 400 个局域和公众数据库中大量不同种类的生命科学类数据。序列数据搜索包括 FASTA、NCBI BLAST 和 WU-BLAST 序列同源性和相似性对比工具。其他还包括蛋白质功能分析、进化树分析、大分子结构分析与多维显示等。

在应用数据库搜索序列时，有两种常见的序列文件格式，如图 1.9 和图 1.10 所示。

GenBank 序列文件中包含了一个基因的每个记录，出现在该图中的不同列中。属于平面文件格式，应用于文字处理计算机语言方面。域的名称显示在前面，完整的记录非常长，因此在这里只显示了顶部。

```
▪ LOCUS       RATOBESE     539 bp ss-mRNA         ROD       23-SEP-1995
▪ DEFINITION  Rat mRNA for obese.
▪ ACCESSION   D49653
▪ KEYWORDS    .
▪ SOURCE      Rattus norvegicus (strain OLETF, LETO and Zucker, ) differentiated
▪             adipose cDNA to mRNA.
▪   ORGANISM  Rattus norvegicus
▪             Eukaryotae; mitochondrial eukaryotes; Metazoa; Chordata;
▪             Vertebrata; Sarcopterygii; Mammalia; Eutheria; Rodentia;
▪             Sciurognathi; Myomorpha; Muridae; Murinae; Rattus.
▪ REFERENCE   1  (bases 1 to 539)
▪   AUTHORS   Murakami,T. and Shima,K.
▪   TITLE     Cloning of rat obese cDNA and its expression in obese rats
▪   JOURNAL   Biochem. Biophys. Res. Commun. 209, 944-952 (1995)
▪   STANDARD  full automatic
▪ COMMENT     Submitted (10-Mar-1995) to DDBJ by:
▪             Takashi Murakami
▪             Department of Laboratory Medicine
▪             School of Medicine
▪             University of Tokushima
▪             Kuramotocho 3-chome
▪             Tokushima 770
▪             Japan
▪             Phone: +81-886-33-7184
▪             Fax:   +81-886-31-9495.
```

图 1.9　序列文件格式 GenBank

```
>gi|995614|gb|D49653|RATOBESE Rat mRNA for obese.

    CCAAGAAGAAGAAGACCCCAGCGAGGAAAATGTGCTGGAGACCCCTGTGCCGGTTCCTGTGGCTTTGGTCC
    TATCTGTCCTATGTTCAAGCTGTGCCTATCCACAAAGTCCAGGATGACACCAAAACCCTCATCAAGACCATTG
    TCACCAGGATCAATGACATTTCACACACGCAGTCGGTATCCGCCAGGCAGAGGGTCACCGGTTTGGACTTCA
    TTCCCGGGCTTCACCCCATTCTGAGTTTGTCCAAGATGGACCAGACCCTGGCAGTCTATCAACAGATCCTCAC
    CAGCTTGCCTTCCCAAAACGTGCTGCAGATAGCTCATGACCTGGAGAACCTGCGAGACCTCCTCCATCTGCT
    GGCCTTCTCCAAGAGCTGCTCCCTGCCGCAGACCCGTGGCCTGCAGAAGCCAGAGAGCCTGGATGGCGTCC
    TGGAAGCCTCGCTCTACTCCACAGAGGTGGTGGCTCTGAGCAGGCTGCAGGGCTCTCTGCAGGACATTCTTC
    AACAGTTGGACCTTAGCCCTGAATGCTGAGGTTTC
```

图 1.10　序列文件格式 FASTA

FASTA 序列文件中包含了 gi 号码、GenBank 检索号码、LOCUS 名称以及 GenBank 记录中的 DEFINATION 字段。第一行(>)表示一个新的序列文件的开始，为标记符。后面可以加上文字说明、gi 号码、GenBank 检索号码、LOCUS 名称等信息。第二行序列为 DNA 或蛋白质的标准符号。通常核苷酸符号大小写均可，而氨基酸则一般用大写字母。不过，有些程序对大小写有明确的要求，使用时需要注意。一般每行 60～80 个字母。

1.5.2　基准数据库

这类数据库中包含的数据一般是真实的、已知结构的蛋白质序列，根据手工比对和反复验证得到的标准结果，通常应用于测试或

衡量比较不同生物信息学软件的性能。例如以下的各数据库或软件：BAliBASE——测试蛋白质多序列比对的准确性；GAGE (genome assembly gold-standard evaluations)——高通量测序结果用于组装基因组，测试组装出来的正确率；CASP(critical assessment of protein structure prediction)/CAFASP——预测蛋白质结构；CAPRI(critical assessment of prediction of interactions)——预测蛋白质相互作用/结合；CAGI (critical assessment of genome interpretation)——预测基因组上的变异会对生物的表型产生什么影响；DREAMchallenges(http://dreamchallenges.org/)——多种不同生物信息学任务的比拼，如预测选择性剪接。

1. 基准比对数据库 BAliBASE

对于绝大多数多序列比对算法来说，很难从理论上给出所得结果与优化比对之间的偏差。当前多序列比对算法众多，且生物序列之间的进化关系也相当复杂，每种序列比对算法都有它相对适用的范围，并非对所有序列都有效。每一种新算法被提出时，作者都要选定一些数据集，与已存在的比对算法进行准确性的比较，这是评价一个算法优劣的方法。但由于每一种算法都有它自己的特性，因此由作者自己选定数据集与其他算法相比显然对其他算法是不公平的。

为了统一评判各种多序列比对算法的有效性，法国生物细胞分子基因组研究所(Institut de Génétique et de Biologie Moléculaire et Cellulaire，IGBMC)的 Thompson 等于 1999 年构建了由真实蛋白质序列组成的基准(benchmark)比对数据库 BAliBASE。该数据库中共存储了 1000 多条真实的蛋白质序列，构成了 142 组参考比对。这些参考比对是基于相应蛋白质的三维结构而确定的，其可靠的比对区域被标注为核心块。这些都已被多个程序证明且通过手工修正了的高质量的比对结果，并且具有良好的文档说明。这些参考比对又根据比对的特点，如序列的长度、相似性以及插/失空位数量

及位置等因素而被划分成五类，涵盖了多序列比对的绝大多数问题。由于 BAliBASE 不是为某一具体比对算法而精心设计的，因此它相对客观，是目前用于评价多序列比对算法有效性的一个通用平台。

2001 年 Bahr 等又在第一版的 BAliBASE 的基础上对原有的参考比对做了进一步修正，并新添加另外三类基准比对共计 1000 多条序列，推出了 BAliBASE2.01。2005 年 Thompson 等又改进了第二版，推出 BAliBASE3.0，在原有的基础上又增加了更多的序列。

一般情况下，常用于测试比对算法的还是 BAliBASE 中的前五类参考比对。

第一类参考比对是由少量的等长序列构成。任意两个序列间相同残基的百分比，即一致率(ID%)要在某一具体范围内，并且不存在大范围的插失，每一比对中都含有 3～7 条序列。

第二类参考比对是在一个蛋白质家族(序列间的亲缘关系较近>25%ID)序列中加入三条“孤儿”序列(“orphan”sequence，家族中亲缘关系较远<20%ID 的成员，但享有共同的折叠)。每一比对中至少含有 15 条近亲序列和 3 条“孤儿”序列。

第三类参考比对中，每一比对至少由 4 个不同家族的蛋白质构成，来自不同家族的任意两个序列相同残基的百分比<25%ID。

第四类参考比对中包含具有大量的 N/C 终端扩展的序列。

第五类参考比对中包含具有大量的内部插/失的序列。

更具体的 BAliBASE 每一类比对数目如表 1.4 所示。

BAliBASE 中提供了两个不同的评分分值 SPS 和 TCS 及程序，分别用于评价与 BAliBASE 中参考比对进行比较的一个测试比对算法的质量，应用该程序包可以直接计算 SPS 和 TCS 分值，如图 1.11 所示。SPS 和 TCS 分值越高，多序列比对算法越好。这两个评分标准将在本书第 4 章中详细介绍。

表 1.4　BAliBASE 数据库

类别	短序列 (<100 个残基)	中等序列 (200～300 个残基)	长序列 (> 400 个残基)
类 1：长度相似的等距离序列			
子类 1：(<25%的一致率)	7	8	8
子类 2：(20%～40%的一致率)	10	9	10
子类 3：(>35%的一致率)	10	10	8
类 2：带孤儿的家族	9	8	7
类 3：等距离的分歧家族	5	3	5
类 4：N/C 终端延展	12		
类 5：内部空位插入	12		

```
with reference alignment in ../RV11/BB11001.msf

          1aab_   ---GKGDPKKPRGKMSSYAFFVQTSREEHKKKHPDASVNFSEFSKKCSER
         1j46_A   ------MQDRVKRPMNAFIVWSRDQRRKMALENPRMRN--SEISKQLGYQ
         1k99_A   MKKLKKHPDFPKKPLTPYFRFFMEKRAKYAKLHPEMSN--LDLTKILSKK
         2lef_A   --------MHIKKPLNAFMLYMKEMRANVVAESTLKES--AAINQILGRR

                  ........1111111111111111111111111..000011111111111

          1aab_   WKTMSAKEKGKFEDMAKADKARYEREMKTYIPPK---GE-------
         1j46_A   WKMLTEAEKWPFFQEAQKLQAMHREKYPNYKYRP---RRKAKMLPK
         1k99_A   YKELPEKKKMKYIQDFQREKQEFERNLARFREDH---PDLIQNAKK
         2lef_A   WHALSREEQAKYYELARKERQLHMQLYPGWSARDNYGKKKKRKREK

                  1111111111111111111111111111111111...00.......

     SP score= 0.961

     TC score= 0.920
auto ../BB11001_org.msf 0.961 0.920
```

图 1.11　BAliBASE 自带 SPS 和 TCS 程序包 baliscore 运行界面

2. 其他蛋白质序列比对基准数据库

在应用基准数据库测试评估多序列比对算法性能时，除了

BAliBASE 数据库，还有其他的蛋白质结构数据库也被作为基准数据库，如 HOMSTRAD 同源结构比对数据库(http://www-cryst.bioc.cam.ac.uk/homstrad/)、SMART 注释了的蛋白质结构域序列(http://smart.embl-heidelberg.de/)、Pfam 蛋白质家族数据库、SCOP 由专家参与的蛋白质结构分类数据库、SABmark 序列比对基准(http://bioinformatics. vub.ac.be)、Prefab3.0、Rose version 1.3 等。

1.6　多序列比对常用工具

1.6.1　搜索工具

一般在多序列比对之前，首先要在数据库中搜索相对应的序列，这里主要介绍 EBI 的 FASTA 工具和 NCBI 的 BLAST 工具，它们是当前最常用的两大数据库搜索工具。

1. FASTA 工具

FASTA 是 FAST-ALL 的缩写，是由 Lipman 和 Pearson 于 1985 年提出的。其基本思路是：

(1) 识别与待查序列相匹配的很短的序列片段，称为 k-tuple。使用者可以改变 ktup 值，一般规定蛋白质序列的 ktup 默认值是 2，DNA 序列的 ktup 默认值是 6。

(2) 运算是寻找与最初识别的单词匹配的扩展，试图找到序列的无空位联配，该联配含有高密度的最初识别的单词匹配，然后再把这些联配加入到高分值的有空位的联配中。最后在识别了序列间的高分值联配后，通过动态规划联配全部序列高分区域，得出最终联配及其分值。

FASTA 可用于核酸和蛋白质序列的快速序列比对数据库搜索。其版本在不断更新升级，可以下载使用，也可以在线进行比对，最

新版本是 FASTA3 软件包，主要包括 FASTA、FASTF、FASTS、FASTX/Y、TFASTX/Y，如表 1.5 所示。

表 1.5　FASTA3 软件包相关内容

程序	查询序列类型	数据库类型
FASTA	DNA、蛋白质	DNA 蛋白质
FASTX、FASTY	DNA	蛋白质
TFASTA	蛋白质	DNA
TFASTX、TFASTY	蛋白质	DNA
FASTS、TFASTS	一系列多肽片段	蛋白质 DNA
FASTF、TFASTF	有序多肽混合物	蛋白质 DNA

随着各生物数据库中序列数量的快速增长，FASTA 工具的搜索速度越来越不能满足用户的要求，因此它逐步被一种速度更快的搜索工具 BLAST 所替代。

2. BLAST 工具

BLAST 是 Basic Local Alignment Search Tool 的缩写，是 Altschul 于 1990 年提出的。其基本思路是：

(1) 寻找打分比某一特定阈值(T)高且长度是 W 的单词。蛋白质序列的 W 默认值是 3，DNA 序列的 W 默认值是 11。使用者可以设置 W 和 T 值，但一般都使用默认值。

(2) 寻找与最初识别的单词匹配的扩展。BLAST 将个别单词匹配扩展，直到联配总分值从最高值跌落一段数量，产生无空位的联配。改进后的 BLAST 程序允许空位的插入。

BLAST是当前应用最广泛的序列相似性搜索工具，研究BLAST的最初目的是改善FASTA算法性能，通过寻找更小更好的热点，提高计算速度。为了进一步提高数据库搜索速度，BLAST增加了限制，即在序列的局部比对中不包括空缺字符。BLAST包含五个程序和若干个相应的数据库，分别针对不同的查询序列和要搜索的数据库类型，如表1.6所示。其中翻译的核算库指搜索比对时会把核酸数据按密码子所有可能的阅读框架转换成蛋白质序列。

表1.6　BLAST软件包相关内容

程序	查询序列类型	数据库类型	简述
BLASTP	蛋白质	蛋白质	适合寻找具有远源进化关系的匹配序列
BLASTN	DNA	DNA	适合寻找分值较高的匹配，不适合远源关系
BLASTX	DNA(翻译)	蛋白质	适合新DNA序列和EST序列的分析
TBLASTN	蛋白质	DNA(翻译)	适合寻找数据库中尚未标注的编码区
TBLASTX	DNA(翻译)	DNA(翻译)	适合分析EST序列

1.6.2　常用的在线多序列比对工具

通过数据库搜索工具收集待测序列后，接下来要进行多序列比对，有很多学者根据多序列比对的原理开发了非常方便好用的比对工具，如MAFFT、CLUSTALW、T-COFFEE等，应用这些比对工具能快速地得到较好的比对结果，成为当前多序列比对最常用的比对手段。在EBI官网中(http://www.ebi. ac.uk/Tools/msa/)提供了

这些常用比对工具对应的开源在线多序列比对工具，其中有 Clustal Omega、MAFFT、T-Coffee、MUSCLE 等工具，界面如图 1.12 所示。相对于线下工具，这些在线比对工具的版本更新及时，更为实用。

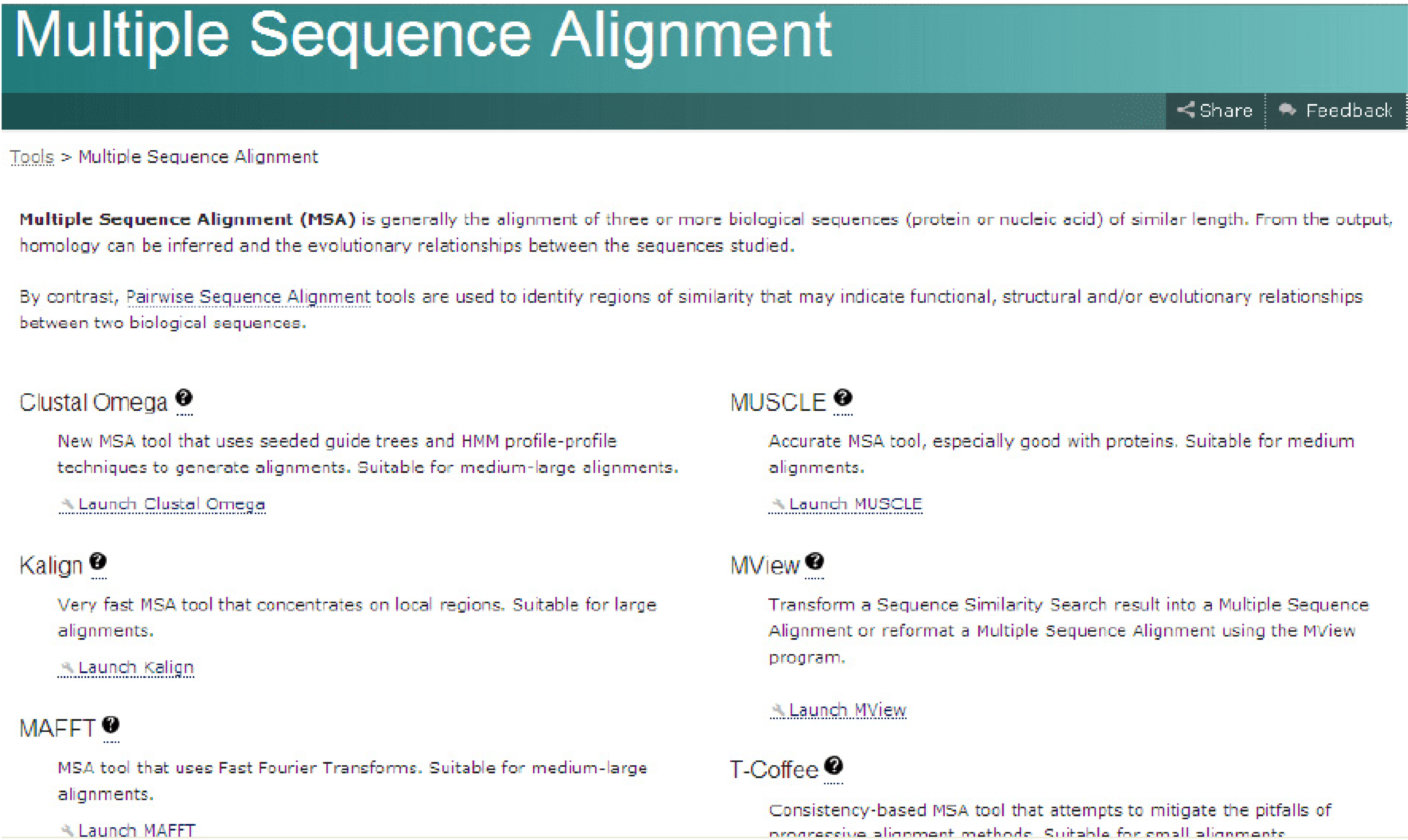


图 1.12 常用比对工具

图 1.13 所示为 MAFFT 工具进行在线序列比对的示例。

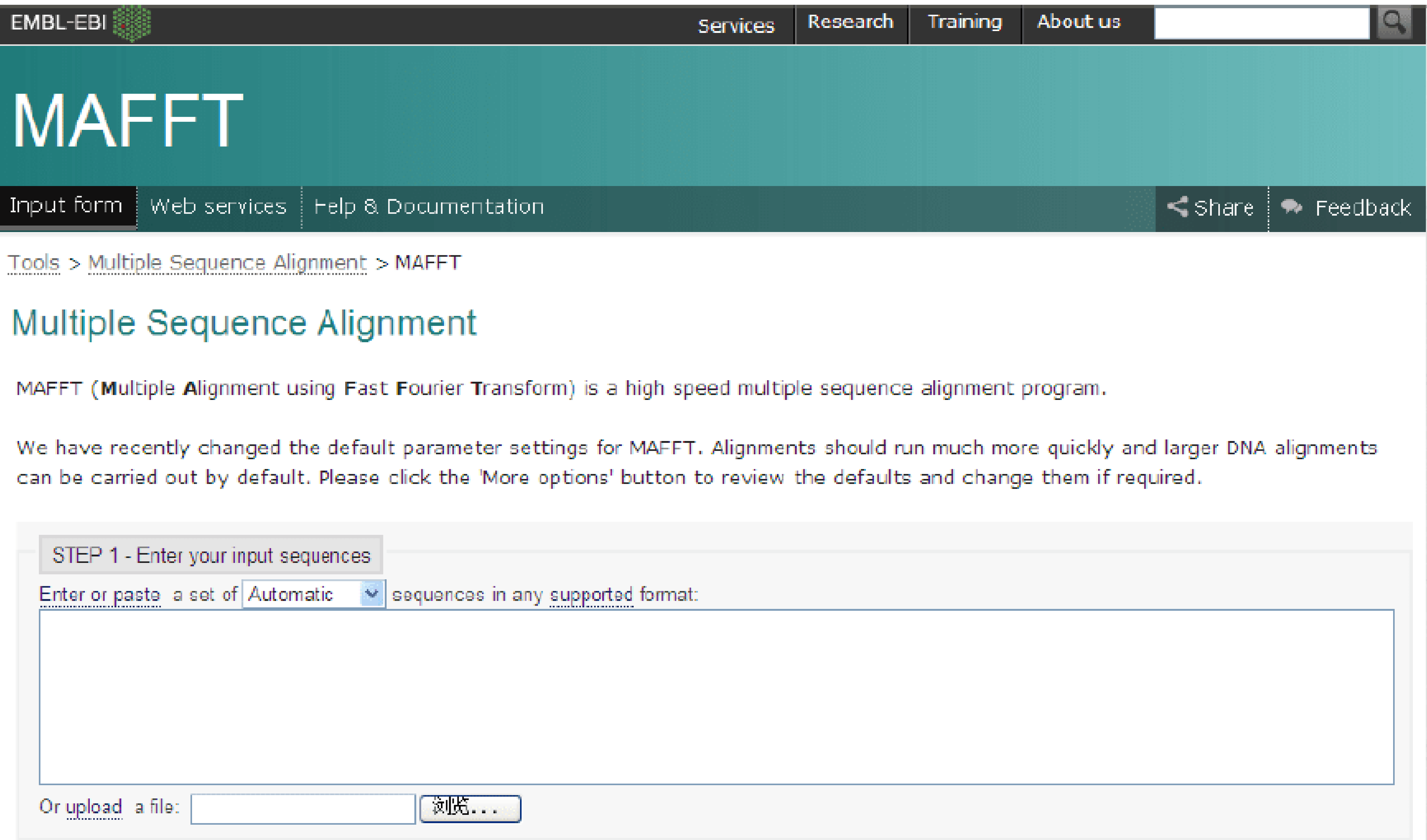


图 1.13 MAFFT 多序列比对工具

在 STEP1 中粘贴要比对的序列，或者单击“浏览”按钮直接选择要比对的序列，如图 1.14 所示。

STEP 2 - Set your Parameters
OUTPUT FORMAT　Pearson/FASTA
The default settings will fulfill the needs of most users and, for that reason, are not visible.
More options... (Click here, if you want to view or change the default settings.)
STEP 3 - Submit your job
Be notified by email (Tick this box if you want to be notified by email when the results are available)
Submit
If you plan to use these services during a course please contact us.

图 1.14　STEP1

在 STEP2 中选择要输出的格式。

在 STEP3 中单击 Submit 按钮，即出现序列比对的结果，如图 1.15 所示。

MAFFT
Input form　Web services　Help & Documentation　Share　Feedback
Tools > Multiple Sequence Alignment > MAFFT
Results for job mafft-I20160509-013128-0883-88146179-pg
Alignments　Result Summary　Guide Tree　Phylogenetic Tree　Submission Details
Download Alignment File　Send to ClustalW2_Phylogeny

```
>1aab_
GK-------GDPKKPRGKMSSYAFFVQTSREEHKKKHPDASVNFSEFSKKCSERWKTMSAK
EKGKFEDMAKADKARYEREMK------TYIPPKG-------E
>1j46_A
MQ-------DRVKRP---MNAFIVWSRDQRRKMALENPRMRN--SEISKQLGYQWKMLTEA
EKWPFFQEAQKLQAMHREKYP------NYKYRPRRKAKMLPK
>2lef_A
MH---------IKKP---LNAFMLYMKEMRANVVAESTLKES--AAINQILGRRWHALSRE
EQAKYYELARKERQLHMQLYPGWSARDNYGKKKKRKRE---K
>1k99_A
MKKLKKHPDFPKKP---LTPYFRFFMEKRAKYAKLHPEMSN--LDLTKILSKKYKELPEK
KKMKYIQDFQREKQEFERNLA------RFREDHPDLIQNAKK
```

图 1.15　STEP3

另外，NCBI 也有在线蛋白多序列比对工具 COBALT(http://www.ncbi.nlm.nih.gov/tools/cobalt/re_cobalt.cgi)，其界面如图 1.16 所示。

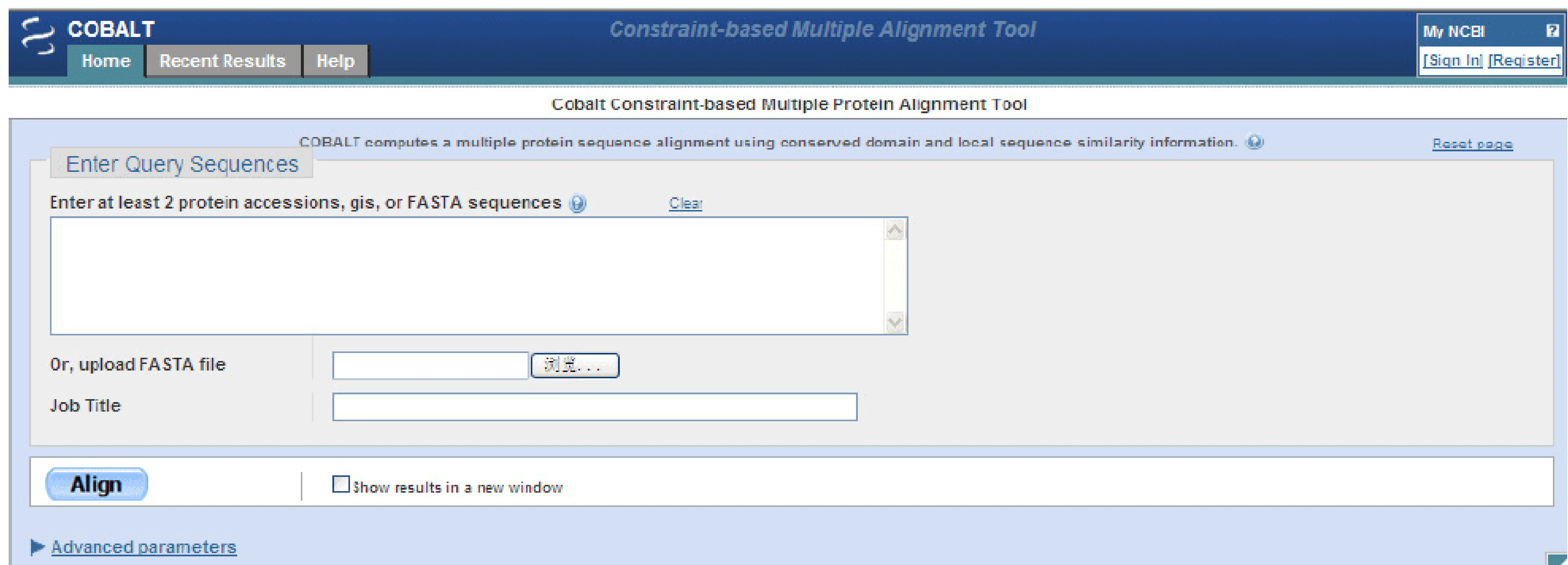


图 1.16　COBALT 多序列比对工具

1.7　多序列比对的应用

在 GenBank 数据库中记录了大量基因或基因组测量的数据，利用这些数据就可以进行多序列比对的计算与分析。下面介绍几个典型的分析与应用例子。

1. 线粒体基因组的比对分析与应用

线粒体在真核生物细胞内普遍存在，且不同生物体的线粒体基因组的长度非常接近。在 GenBank 数据库内保存了上千种线粒体的基因组数据，对线粒体基因组做 MSA 计算与分析是了解这些生物体进化过程的重要手段。

2. 关于流行病基因组的比对分析与应用

流行疾病病毒基因组的类型很多，对这些病毒基因组作比对分析可以了解这些病毒基因组在传播过程中的变异状况。对流行病病毒基因组的比对分析有：

(1) 关于 SARS 病毒的比对分析。在 GenBank 数据库中记录的 SARS 病毒组的 MSA 规模是 108×30kbp。通过 SARS 病毒基因组的 MSA 可以了解到从果子狸到人，以及人的早中晚期的基因突变状

况，尤其是在从果子狸到人，以及人在早中晚期传播时发生较大规模爆发时的基因突变特征。

(2) 关于 HIV-Ⅱ病毒的比对分析。HIV-Ⅱ病毒是一种艾滋病的病毒，它的病毒数据不仅年代长，而且分布地区广，测量所得到的数据还有潜伏期的长短等问题，因此对它们的分析要复杂得多。对 HIV-Ⅱ病毒有专门的数据库，对它们作比对分析的问题也很多，如传播的途径与方式问题；从境外到国内不同地区的传播过程的途径分析以及不同的传播方式(如性交、吸毒、血液等)的分析等；HIV-Ⅱ病毒的类型与测定时间的分析。对 HIV-Ⅱ病毒的分析还可以分解为对多种不同类型的 HIV-Ⅱ病毒，以及测定时病毒潜伏期时间长短的分析等。

(3) 其他类型的流行病，如流感、禽流感等。

1.8　其他说明

1.8.1　多序列比对算法存在的问题

多序列比对算法主要存在以下三大问题：

(1) 多序列的比对问题。序列比对问题目前所存在的问题就是优化双序列比对算法应用于多序列比对，该问题在计算生物学与生物信息学中仍被列为未解决的重大问题或非易计算问题，有的文献把多序列比对列为 NP 完全问题，计算复杂度为双指数问题，也就是计算复杂度为 $O(2^{m+n})$，其中 m 为比对序列的重数，n 为序列长度。因此目前由以上序列比对算法改良的多序列比对只能在小规模上进行，这与目前所出现的庞大数据是极不相称的，如何构造多序列比对的快速算法是计算机生物学与生物信息学中的重大问题。

(2) 比对结果的分析问题。同源序列比对的根本目标是确定它们的进化演变关系，在生物学界常常通过序列比对来构造进化树，

并由此来确定它们的进化关系。但是，比对与进化的关系到底如何，如何由大量序列的比对结果来构造进化树的逻辑过程是生物学家所关心的问题。一个典型的问题是，在现有的比对理论中，把寻找罚分最小(或得分最多)的比对结果看作序列突变与进化的结果，但是求罚分最小的比对结果与序列突变的结果并不完全一致。因此，如何由序列比对来确定序列的突变与进化关系，如何建立它们变化关系的数学模型与分析规则是序列比对理论中不可缺少的一个重要部分。

(3) 不同比对算法的效果分析问题。目前无论双序列还是多序列的比对都有大量算法的出现，另外，对这些比对算法结果也有许多度量性的指标进行评价与考核，如计算复杂度(时间复杂度与空间复杂度)、比对相似度、搜索相似度等，因此对这些不同的比对算法如何建立它们的考核体系尤为重要。目前，对生物信息学的考核还是以进行实际测试计算为最一般的考核方法，要对所设计的比对算法的各项度量指标做出理论上的说明是一个十分困难的问题，因为这涉及序列突变的总体或局部模型问题，这是一个十分复杂的问题，它不仅序列数据庞大，而且突变现象千变万化，所以不可能用一种或几种模型就能把它们概况说明。

1.8.2　多序列比对算法的运算指标

近几年内有许多多重序列的比对问题十分活跃，许多算法、软件包与比对结果大量出现，这些算法或软件包都是在次优解的意义下实现计算。因此，对于不同的多重序列比对算法的好坏需要比较其多种运算指标。多重序列比对算法的主要运算指标有：

(1) 比对规模。就是对比对序列 A 的长度与重数的要求，MSA 算法已基本上实现了无规模的限制。

(2) 运算速度。理想状态的运算速度就是实现 $O(mn)$ 的比对计算复杂度，其中 m 和 n 分别是多重序列的平均长度与重数，MSA 算

法可基本上实现该计算复杂度。

(3) 优化指标的讨论。优化指标是多重序列比对算法中的一个关键问题，建立多重序列比对的优化指标体系实际上是关系到如何理解多重序列比对的优化问题。

1.8.3　多序列比对算法的展望

近年来，随着人们对生物序列认识的逐渐深入，越来越多的蛋白质三维结构及其功能被人们所认识，新的生物序列的结构信息、功能信息、进化关系等也相应加入到序列比对模型中。多序列比对方法的研究现状表明，该领域的相关研究十分活跃，并且取得了巨大的进展，其方法也趋于成熟，但随着大量生物数据的不断加入，多序列比对仍然是进行生物序列分析的基础，在此领域仍有许多问题值得进一步的探索。

(1) 建立更能准确反映生物数据特性的多序列比对数学模型，使得比对结果更加精确。

(2) 改进现有的多序列比对算法，加快问题的求解速度。

(3) 结合渐进比对法和迭代比对法的优点，提出能够克服“局部最小化”问题的快速多序列比对算法。

(4) 比对算法的并行化。由于计算资源的限制和问题求解规模的逐步增大，因此需要实现算法的并行化。

(5) 算法性能分析。由于多序列比对问题的复杂性较高，算法性能分析与评价方法的研究对于算法的改进和优化具有非常重要的意义，因此需要研究不同条件和背景下的算法性能分析方法。

1.9　本章小结

为了详细描述生物多序列比对问题，本章从一些相关的概念

和知识引入，介绍了多序列比对的基本原理、方法、常用数据库等内容，最后介绍了多序列比对的常用工具和应用。

参 考 文 献

[1] 张春霆. 生物信息学的现状与展望[J]. 世界科技研究与发展，2000，22(6)：17-20.

[2] Chuong B, Do, Katoh K. Protein multiple sequence alignment[J]. Humana Press, 2008,484(12): 379-413.

[3] 邹权，郭茂祖，韩英鹏. 多序列比对算法的研究进展[J]. 生物信息学，2010(4)：311-315.

[4] Notredame C. Recent progress in multiple sequence alignment: a survey[J]. Pharmacogenomics, 2002, 3(1): 131-144.

[5] 张鹏帅，霍红卫. 多序列比对问题的粒子群优化算法求解[J]. 计算机工程与应用，2005，41(18)：84-87.

[6] Gusfield D. Algorithms on strings, trees and sequences: computer science and computational biology[M]. Cambridge university press, 1997.

[7] Thompson J D, Higgins D G, Gibson T J. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice[J]. Nucleic acids research, 1994, 22(22): 4673-4680.

[8] 程灏. 混合遗传算法在图着色及 MSA 问题中的应用[D]. 西安电子科技大学，2006.

[9] 林秋利. 基于进化算法和量子计算的多序列比对方法研究[D]. 西安电子科技大学，2006.

[10] Dan D B, Kececioglu J. Parameter advising for multiple sequence alignment[J]. Bmc Bioinformatics, 2015, 16(1): 516-518.

[11] Gondro C, Kinghorn B P. A simple genetic algorithm for multiple sequence alignment[J]. Genetics and Molecular Research, 2007, 6(4): 964-982.

[12] Wang L, Jiang T. On the complexity of multiple sequence alignment[J]. Journal of Computational Biology, 1994(1): 337-348.

[13] 张敏. 生物信息学中多序列比对等算法的研究[D]. 大连理工大学，2005.

[14] Notredame C, Higgins D G, Heringa J. T-COFFEE: a novel method for fast andaccurate multiple sequence alignments[J]. J.Mol.Evol., 2000, 302(1): 205-217.

[15] 韦树烽，刘羽，蒋财运. 基于 GPU 的遗传退火多序列比对并行研究[J]. 计算机工程与设计，2014，35(4)：1247-1252.

[16] Kaya M, Sarhan A, Alhajj R. Multiple sequence alignment with affine gap by using multi-objective genetic algorithm[J]. Computer methods and programs in biomedicine, 2014, 114(1): 38-49.

[17] Kwan M, Xiao N, Ding G. Assessing Activity Pattern Similarity with Multidimensional Sequence Alignment Based on a Multiobjective Optimization Evolutionary Algorithm[J]. Geographical Analysis, 2014, 46(3): 297–320.

[18] Notredame C, Higgins D G. SAGA: sequence alignment by genetic algorithm[J]. Nucleic acids research, 1996, 24(8): 1515-1524.

[19] 葛宏伟，梁艳春. 基于隐马尔可夫模型和免疫粒子群优化的多序列比对算法[J]. 计算机研究与发展，2006，43(8)：1330-1336.

[20] Chen W, Liao B, Zhu W, et al. An ant colony pairwise alignment based on the dot plots[J]. Journal of Computational Chemistry, 2009, 30(1): 93–97.

[21] Silva F J M, Sánchez Pérez J M, Gómez Pulido J A, et al. AlineaGA—a genetic algorithm with local search optimization for multiple sequence alignment[J]. Applied Intelligence, 2010, 32(2): 164-172.

[22] Silva F J M, Sánchez Pérez J M, Antonio J, et al. An evolutionary approach for performing multiple sequence alignment[C]. WCCI 2010 IEEE World Congress on Computational Intelligence, Barcelona, Spain, July. 2010: 18-23.

[23] Rodríguez M A V. Optimizing Multiple Sequence Alignment by Improving Mutation Operators of a Genetic Algorithm[J]. Intelligent Systems Design and Applications, 2009.ISDA '09.Ninth International Conference on, 2009: 1257-1262.

[24] Chakrabarti T, Saha S, Sinha D. DNA Multiple Sequence Alignment by a Hidden Markov Model and Fuzzy Levenshtein Distance based Genetic Algorithm[J]. International Journal of Computer Applications, 2013: 73.

[25] Naznin F, Sarker R, Essam D. Progressive Alignment Method Using Genetic Algorithm for Multiple Sequence Alignment[J]. Evolutionary Computation, IEEE Transactions on, 2012, 16(5): 615-631.

[26] 张鑫源，胡晓敏，林盈. 遗传算法和粒子群优化算法的性能对比分析[J]. 计算机科学与探索，2014(1)：90-102.

[27] Zou Q, Shan X, Jiang Y. A Novel Center Star Multiple Sequence Alignment Algorithm Based on Affine Gap Penalty and K-Band[J]. Physics Procedia, 2012(33): 322-327.

[28] 杨锡南，孙啸. 生物信息学中基因数据可视化[J]. 计算机与应用化学，2001(18)：403-410.

[29] 贺向敏，周根宝. 基于遗传算法的多序列比对算法研究[D]. 内蒙古农业大学，2010.

[30] 张琎，张远. 基于 GC-GM 的多序列比对穷举遗传算法[J]. 计算机应用，2010，30(1)：146-149.

[31] 刘立芳，霍红卫，王宝树. PHGA-COFFEE：多序列比对问题的并行混合遗传算法求解[J]. 计算机学报，2006，29(5)：727-733.

[32] Fan H, Wu R, Liao B, et al. An Improved Genetic Algorithm

for Multiple Sequence Alignment[J]. Journal of Computational and Theoretical Nanoscience, 2012, 9(10): 1558-1564.

[33] Thomsen R, Boomsma W. Multiple sequence alignment using SAGA: investigating the effects of operator scheduling, population seeding, and crossover operators[M]. Applications of Evolutionary Computing. Springer Berlin Heidelberg, 2004: 113-122.

[34] Yanping Lv, Shaozi Li, Changle Zhou, etal. Improved Genetic Algorithm for Multiple Sequence Alignment Using Segment Profiles (GASP). Computer Science, 2006, 4093(2006), 388-395.

[35] 泽瓦勒贝 M，鲍姆 J O. 理解生物信息学[M]. 北京：科学出版社，2012.

[36] 陈铭. 生物信息学[M]. 北京：科学出版社，2012.

[37] Edgar R C. MUSCLE: multiple sequence alignment with high accuracy and high throughput[J]. Nucleic Acids Research, 2004, 32(5): 1792-1797.

[38] 杨晶. 生物计算[M]. 北京：科学出版社，2010.

[39] 刘立芳. 生物信息学中的多序列比对与模体识别问题研究[D]. 西安电子科技大学，2006.

[40] 徐小俊. 群智能优化算法在多序列比对中的应用[D]. 陕西师范大学，2011.

第2章　进化算法和最优化理论

2.1　进化算法

进化算法是一类借鉴生物界自然选择和自然遗传机制的随机搜索算法，主要包括以下几种不同的方法：遗传算法(GA)和遗传规划(GP)、进化策略(ES)、进化规划(EP)。

粒子群优化(PSO)算法和量子粒子群优化(QPSO)算法是近年来发展起来的一种新的进化算法。PSO 算法和 QPSO 算法属于进化算法的，和遗传算法相似，它也是从随机解出发，通过迭代寻找最优解，它们也是通过适应度来评价解的品质。但是它们比遗传算法规则更为简单，它没有遗传算法的“交叉”(crossover)和“变异”(mutation)操作。它通过追随当前搜索到的最优值来寻找全局最优。

2.1.1　遗传算法

遗传算法是一种借鉴基因遗传机理和达尔文适者生存的自然选择原则，通过模拟群体自然进化过程的随机搜索算法。1975 年，Holland 出版了专著《自然系统和人工系统的适配》。他在该书中首次阐述了遗传算法的基本理论与实施方法。遗传算法在解决复杂的全局优化问题(如多峰目标函数或不规则搜索空间)方面，因其具有鲁棒性，适于并行计算，很快就得到了广泛的应用。

为了使用遗传算法，首先需要对问题进行编码。编码是对问题可行解的遗传表示，编码的好坏直接影响到遗传算法应用的成败。通常采用固定长度的二进制编码，通过编码组成初始群体后，遗传

操作的任务就是对群体的个体按照它们对环境适应度(适应度评估)施加一定的操作，从而实现优胜劣汰的进化过程。从优化搜索的角度而言，遗传操作可使问题的解一代又一代地优化，并逼进最优解。

遗传操作包括以下三个基本遗传算子(genetic operator)：选择、交叉和变异。

(1) 选择：染色体的适应度越高，其被选择的机会就越多，可采用轮盘赌方式来实现选择。选择过程的目的是从当前群体中选出优良的染色体。

(2) 交叉：对被选出的优良染色体进行交叉操作，交叉操作是组合父母有价值的遗传信息，具有改变遗传模式的功能，它由交叉概率调控。

(3) 变异：交叉操作后再进行变异操作，变异操作的目的是挖掘种群中个体的多样性，克服可能陷入局部最优解的弊病，它由变异概率调控。

2.1.2　遗传规划

遗传规划(GP)的思想是由 Stanford 大学的 Koza 在 20 世纪 90 年代初提出，并于 1991 年出版了专著 *Genetic Programming*。

GP 算法每一代群体中的个体均采用一种动态的树状结构(如图 2.1 所示)。树的结点由叶结点和函数集组成。叶结点由输入变量和常量构成。而函数集用于将叶结点连接起来，形成一个表达式。图 2.1 形成的表达式为 $(x-2)+\cos(y)*3.5687$ 。每个树状结构对应着一个计算机程序。叶结点中的变量相当于计算机程序的输入变量，而树结构所代表的表达式的值即为计算机程序的输出值。

GP 的基本思想是：随机产生一个适合于给定问题环境的初始群体(初始解)，即问题的搜索空间，构成群体的个体都有一个适应度，依据适者生存原则，用遗传算子处理得到高适应度的个体，产生下一代群体，如此进化下去，直到在某一代得到给定问题的解或近似解。

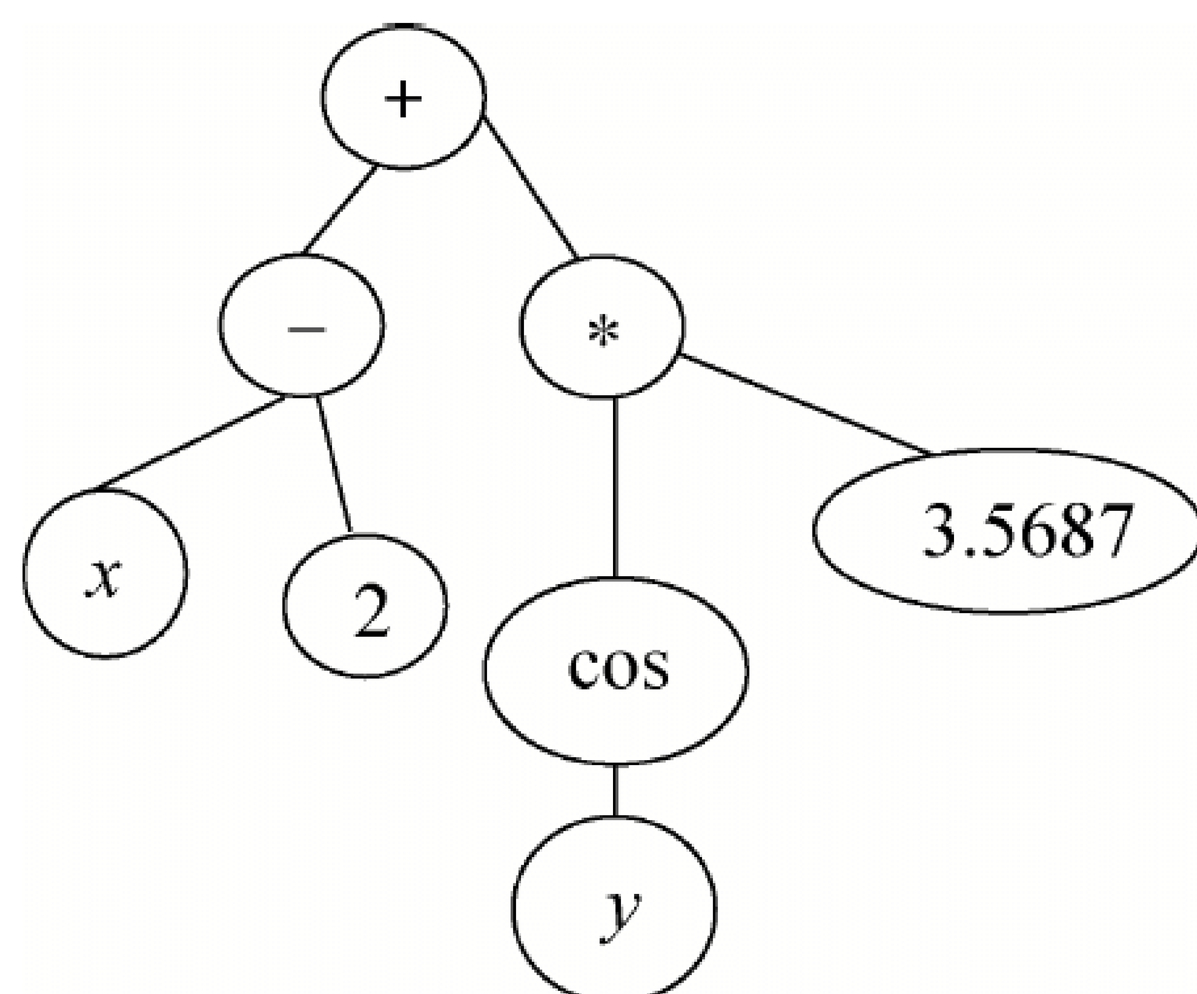


图 2.1　遗传规划的树形结构

遗传规划与标准的遗传算法一样，主要有选择、交叉和变异三个遗传算子。

(1) 选择：选择操作的目的是把当前群体中适应度较高的个体按照某种规则遗传到下一代群体中。一般而言，个体的适应度越高，被选择复制的机会就越大。适应度的选择方法主要有轮盘赌选择法、随机遍历抽样法、局部选择法和锦标赛选择法等。轮盘赌选择法是最基本也是最常用的选择方法。

(2) 交叉：交叉操作的目的是增加群体中的新个体，从而扩大群体的搜索空间。交叉时，每个父代个体随机选择一个交换点，于是便产生一个以交换点为根的子树，该子树包括交换点以下的所有子树，此子树称为交换段。有时一个交换段是一片叶子。将第 1 个父代个体删除其交换段后，再把第 2 个父代个体的交换段插入其交换点处，这样就产生了第 1 个子代个体，同样操作可产生第 2 个子代个体。

(3) 变异：变异的目的是维持群体的多样性。但是遗传规划中变异算子是次要算子。因为一个个体由函数集和终止符集组成，所以变异也分函数变异和终止符变异两种形式。

采用 GP 算法解决问题时需要确定以下五个元素：

(1) 终端结点。由输入变量和常量构成。

(2) 函数结点。由算术运算符(+、−、*、/)、数学函数(sin、cos、tan、exp、log)、布尔运算符、条件运算符等构成。

(3) 适应度(fitness)：适应度评价函数(该性能代表个体解决目标问题的能力)。种群中每个个体都会依照适应度评价函数算出一个适应度值。

(4) 算法控制参数(algorithm controlling parameter)。包括种群的大小，遗传操作如交叉、复制、变异的概率。

(5) 终止条件(terminate condition)。终止条件通常是预先给定的，可以是最大进化代数或要求的最小适应度值。

这五个要素中，前三个决定了算法的搜索空间，而后两个则决定了算法的质量和速度。

2.1.3 进化策略

进化策略是最早出现的一种进化算法，由 Rechenberg 和 Schwefel 于 1964 年为优化物体形状参数而提出。它用传统的实型数表达问题，其表达形式为

$$X^{t+1} = X^{t} + N(0,\sigma) \tag{2.1}$$

式中，X^{t} 是用实数表示的第 t 代个体；X^{t+1} 是用实数表示的第 t+1 代个体；$N(0,\sigma)$ 是独立的随机数，服从正态分布，后者的数学期望为 0，标准差为 σ。

式(2.1)表明，新一代的个体 X^{t+1} 是在父代个体 X^{t} 的基础上添加一个随机量 $N(0,\sigma)$，因此每个个体由 X 及 σ 两个变量决定，是一个二元组(X,σ)。

进化策略中个体的进化主要采用突变，并对随机量的标准差进行修正：

$$\begin{cases} \sigma' = \sigma \bullet \mathrm{e}^{N(0,1)} \\ X' = X + N(0,\sigma') \end{cases} \tag{2.2}$$

式中，(X,σ)为父代个体，(X',σ')为子代个体。

也就是说，新一代 X' 是在上一代的 X 基础上添加一个微小的随机量 $N(0,\sigma')$，后者服从数学期望为 0、标准差为 σ' 的正态分布。新一代的标准差 σ' 又是在上一代标准差 σ 的基础上乘以一个微小的随机量 $\exp(N(0,1))$。

在进化策略中，产生新个体的另一种方法是重组，它相当于遗传算法的交叉。最简单的重组是随机交换两个个体的 X_i 及 σ_i。

在进化策略中，复制隐含在选择中。父代群体所有的 μ 个个体，经过突变、重组后生成 λ 个新个体，然后再从这些群体中，按照适应度大小选择 μ 个优良个体组成下一代群体，从而体现个体在竞争中的优胜劣汰原则。同样，进化策略也是一个反复迭代的过程，它从随机产生的初始群体出发，经过突变、重组(交换)、选择等进化操作，改进群体的质量，逐渐得到最优解。

2.1.4　进化规划

20 世纪 60 年代中期，Fogel 等为有限状态机的演化提出进化规划用来求解预测问题，其基本思想是源于对自然界中生物进化过程的模拟。进化规划与进化策略几乎同时出现，并平行发展，最早的进化策略只采用单个个体，而最早的进化规划则是采用多个个体组成群体共同进化。

进化规划也是用实型数表达问题，其表达形式为

$$X^{t+1} = X^t + \sqrt{f(X^t)} \cdot N(0,1) \tag{2.3}$$

式中，X^t 是用实数表示的第 t 代旧个体；X^{t+1} 是用实数表示的第 t+1 代新个体；$N(0,1)$是独立的随机数，服从(0,1)标准正态分布；$f(X^t)$ 是 X^t 的适应度。

早期的进化规划，只用上式实现个体的不断变化，它相当于突变、后期的进化规划，也仿效进化策略引入方差的调整作用，即

$$\begin{cases} X = X + \sqrt{\sigma} \cdot N(0,1) \\ X' = \sigma + \sqrt{\sigma} \cdot N(0,1) \end{cases} \tag{2.4}$$

在进化规划中，没有重组或交换，但有选择，它从 μ 个父代个体及 λ 个子代个体中择优选取 μ 个优良个体组成下一代群体。

2.1.5　粒子群优化算法

群体智能(swarm intelligent，SI)算法始于 20 世纪 90 年代初，主要是受自然界生物群体所表现出智能现象的启发，通过模拟社会性生物的群体行为，而提出的一种随机优化算法。群体智能的核心是由众多简单个体组成的群体能够通过相互之间的简单合作来实现某一较复杂的功能，完成某一较复杂的任务。所以群体智能可以在没有集中控制并且缺少全局信息和模型的前提下，为解决复杂的分布式问题提供了可能。典型方法有 Dorigo M 提出的蚁群算法以及 Kennedy J 与 Eberhart R.C 提出的粒子群优化算法(PSO)。

蚁群算法(ant colony optimization，ACO)，又称蚂蚁算法，是一种用来在图中寻找优化路径的概率型算法，由 Marco Dorigo 于 1992 年在他的博士论文中提出，其灵感来源于蚂蚁在寻找食物过程中发现路径的行为。可以将蚁群在觅食过程中的生物行为描述如下：设一个蚂蚁群体中有 n 只蚂蚁，单个蚂蚁能将 A 处的食物搬运到巢穴 B 处。每只蚂蚁在运动中会在所经过的路径释放一种特殊的分泌物——信息素，并且通过信息素来实现个体之间的“通信”。发现并且运回食物的蚂蚁 a_1 遇见迎面而来的 a_2，a_1“告诉”a_2 从 A 处搬运食物回巢穴 B 处，即将信息传给对方。在整个过程中，每只蚂蚁的行动是随机、并行进行的。开始的时候，蚂蚁的搬运是无序的，但经过一段时间以后，由于各个个体之间的通信便逐渐形成了有序的搬运、堆放等群体活动。

由此可见，蚁群算法的基本思想是模拟蚂蚁在寻找食物源时，在其走过的路上释放“信息素(随着时间的推移该物质会逐渐挥发)”，选择路径的概率与这条路径上强度成正比。当路径上通过的蚂蚁越来越多时，其留下的信息素轨迹也越来越多，蚂蚁选择该路

径的概率也越高，增加了该路径的信息素强度。强度大的信息素会吸引更多的蚂蚁，从而形成一种正反馈机制。通过这种正反馈机制，蚂蚁最终可以发现最短路径。当蚂蚁巢穴与食物源之间出现障碍物时，蚂蚁不仅可以绕过障碍物，而且通过蚁群信息素轨迹在不同路径上的变化，经过一段时间的正反馈，最终收敛到最短路径上。

自 ACO 算法提出以来，已经成功地用于生产调度、机器人路径规划、通信路由等领域的组合优化问题。

美国社会心理学家 Kenney 和电气工程师 Eberhart 于 1995 年提出了 PSO 算法。主要思想来源于对鸟类群体行为的研究，他们的模型和仿真算法主要利用了生物学家 Heppner 提出的模型。尽管最初的设想是通过仿真鸟群这样的简单社会系统来研究并解释复杂的社会行为，但随着研究的深入，大家发现 PSO 还是一种能有效解决复杂优化问题的技术，它通过群体中粒子间的合作与竞争而产生的群体智能进行指导优化搜索。PSO 与人工生命，特别是进化算法有着极为特殊的联系，其都遵循自然界的进化原则，但比进化算法又更多地保留了基于种群的全局搜索策略。PSO 算法采用简单的速度－位移模型，避免了复杂的遗传操作，同时它特有的记忆功能使其可以动态地跟踪当前的搜索情况并调整搜索策略，具有较强的全局搜索能力和鲁棒性，且不需要借助问题的特征信息。因此，PSO 是一种更高效的并行搜索算法，非常适用于对复杂环境中的优化问题的求解。PSO 算法解决问题是先初始化一组随机解，通过迭代搜寻最优值。在 PSO 算法中，每个优化问题的解看作搜索空间的一只鸟，称为“粒子”。所有的粒子对应着优化问题的适应值，粒子的速度决定其飞行的方向和距离，粒子通过追寻群体中的最优粒子来完成在解空间的搜索。PSO 算法自提出以来，由于其计算简单、易于实现、控制参数少等特点，引起了国内外相关领域众多学者的关注和研究。

在 PSO 算法的改进方面，Kennedy 和 Eberhart 在 1997 年提出的二进制 PSO 算法，为 PSO 算法与遗传算法的性能比较提供了一个有用的方式；其次为了提高算法的收敛性能，Shi 和 Eberhart 在 1998

年对 PSO 算法引入了惯性权重 w，并在进化过程中动态调整惯性权重以平衡全局性和收敛速度，该进化方程被称为标准 PSO 算法(standard PSO，SPSO)；2001 年他们又提出了基于模糊系统的惯性权重动态调整方法；Clerc 于 1999 年在进化方程中引入收缩因子以保证算法的收敛性，同时使得对速度的限制放松，使 PSO 算法具有更好的收敛率。Angeline 于 1998 年和 1999 年借鉴进化计算中的选择和杂交概念，将其引入 PSO 算法中以提高算法的收敛性。为了提高 PSO 算法的全局收敛能力，Suganthan 在标准 PSO 算法中引入空间邻域的概念，将处于同一空间邻域的粒子构成一个子粒子群分别进行进化，并随着进化动态地改变阈值以保证群体的多样性；Kennedy 引入邻域拓扑的概念来调整邻域的动态选择，同时引入社会信念将空间邻域与邻域拓扑中的环拓扑相结合，增加邻域间的信息交流，提高种群的多样性。Lovbjerg 等于 2001 年将遗传算法中的子群体概念引入 PSO 算法中，同时引入繁殖算子以进行子群体的信息交流；Kennedy 于 2004 年从概率统计的角度，将粒子的运动改为正态分布的随机扰动，并采用邻域环拓扑结构来改进 PSO 算法的性能；同年，R.A. Krohling 将演化方程中的加速因子变动方式由原来的均匀随机分布改为正态分布，提高了算法的收敛能力；Riget 等从利用群体的多样性出发，在 PSO 算法中增加了发散行为，较好地提高了算法的全局搜索能力；曾建潮等提出利用控制理论对 PSO 算法进行分析，并从提高算法效率的角度出发，建立了积分环节与振荡环节组成的系统所对应的改进 PSO 算法；他们还分别提出了基于微分模型和模拟退火算法的改进 PSO 算法；高海兵等提出了适用于解决离散问题的广义粒子群优化模型，并将算法成功地应用于 TSP 问题中；窦全胜等从增强粒子群优化能力出发，将模拟退火和分工策略两种机制引入到了 PSO 算法中；刘宇等充分利用粒子速度信息，改变了粒子速度的更新方式，并且引入了精英集团策略提出了简约 PSO 算法；刘洪波等在分析了 PSO 算法的收敛性的基础上利用混沌特性提高种群的多样性和搜索遍历性，提高了

粒子的持续搜索能力；俞欢军等针对 PSO 算法在多峰函数优化中易于早熟的缺点，提出了基于反馈策略的自适应 PSO 算法，提高了算法求解的成功率和精确度。

2.1.6　量子粒子群优化算法

由 Sun 等提出的 QPSO 算法来源于量子空间中的量子模型。量子系统由于态叠加性而具有很强的不确定性，而人类思维也是具有不确定性的，因而用量子模型描述人类思维和智能是合乎逻辑的。关键问题是如何建立一个有效量子模型。

研究表明，聚集性是群体智能最基本的特点。所谓聚集性，就是群体中个体的差异是有限的，不可能趋向无穷大。聚集性是由群体中的个体具有相互学习的特点决定的，个体的学习有以下特点：

(1) 追随性，即个体总是倾向于学习群体中最优的知识。这种性质使个体的差异减小。

(2) 记忆性，即个体在学习过程中，受到自身经验知识的约束，而这种特性使个体差异增加。由于这两个特性，个体在学习过程中同时受到群体最好知识和本身经验知识的影响，通过学习获得一种介于群体最好和个体经验之间的知识。但总体而言，具有这两种性质的学习可使个体间差异减少，群体多样性降低。

(3) 创造性。创造性使个体远离现有知识，使个体的差异扩大，群体多样性增加。

聚集性是趋同和趋异两种趋势共同作用的结果，但趋同的趋势更大，否则就没有聚集性。从算法的角度分析，追随性和记忆性的共同作用代表局部搜索能力，创造性代表全局搜索能力。

在考虑建立量子行为粒子群算法的模型时，决策变量同样用粒子的当前位置表示(用向量 $\boldsymbol{X}$ 表示)，代表个体的当前思维状态；粒子经验中搜索到的具有最好适应值(目标函数值)的位置代表个体经验知识(即 pbest)；当前群体中的具有最好适应值的粒子位置

代表群体最好知识(即 gbest)。聚集性在力学中，用粒子的束缚态来描述。产生束缚态的原因是在粒子运动的中心存在某种吸引势场。为此可以建立一个量子化的吸引势场来束缚粒子(个体)以使群体具有聚集态。处于量子束缚态的粒子可以以一定的概率密度出现在空间任何点，它只要求当粒子与中心的距离趋向无穷时，概率密度趋近 0。因此量子模型的随机性更大，关键问题就是如何建立以及采用何种形式的势能场。

M. Clerc 通过代数和数学分析方法，对 PSO 算法中粒子收敛行为进行了分析。研究表明，粒子 i 的收敛过程以点 $p_i=(p_{i1},p_{i2},\cdots,p_{iN})$ 为吸引子，其坐标为：

$$p_{i,j}(t)=\frac{c_1r_{1,j}(t)P_{i,j}(t)+c_2r_{2,j}(t)G_j(t)}{c_1r_{1,j}(t)+c_2r_{2,j}(t)}\quad(1\leqslant j\leqslant N)\tag{2.5}$$

或者

$$p_{i,j}(t)=\varphi_{i,j}(t)\cdot P_{i,j}(t)+[1-\varphi_{i,j}(t)]\cdot G_j(t)\tag{2.6}$$

式中，

$$\varphi_{i,j}(t)=c_1r_{1,j}(t)/[c_1(t)r_{1,j}(t)+c_2r_{2,j}(t)]\tag{2.7}$$

实际上当 c_1=c_2 时，$\varphi_{i,j}(t)$ 本身就是一个区间(0,1)上均匀分布的随机数，即 $\varphi_{i,j}(t)\sim U(0,1)$。因此在实际计算过程中可以直接由随机数发生器产生，这时式(2.6)写成

$$p_{i,j}(t)=\varphi_{i,j}(t)\cdot P_{i,j}(t)+[1-\varphi_{i,j}(t)]\cdot G_j(t)\ \ \varphi_j(t)\sim U(0,1)\tag{2.8}$$

在量子行为粒子群算法中，采用这个式子。

在收敛过程中，随着速度的减小，粒子 i 不断地接近 p_i 点，最后跌落到 p_i 点。因此在整个过程中，在 p_i 点处实际上存在某种形式的吸引势吸引该粒子，这正是整个群体保持聚集性的原因。但由于在经典的 PSO 系统中，粒子的收敛是以轨道的形式实现的，并且粒子的速度总是有限的，因此在搜索过程中粒子每个迭代步的搜索空间是一个有限的区域，不能覆盖整个可行空间。因此一般的 PSO 算法不能保证以概率 1 收敛到全局最优解，这正是一般

PSO 算法的最大缺陷。而在量子空间中，粒子的聚集性通过在粒子运动中心存在的某种吸引势产生的束缚态来描述，而处于量子束缚态的粒子可以以一定的概率密度出现在空间任何点，满足聚集态的性质的粒子可以在整个可行解空间中进行搜索，但不会发散到无穷远处。

根据以上基本思想，考虑在 p_i 点建立一个吸引势。通过比较发现，δ 势阱场可以产生比较好的效果，于是提出了基于 δ 势阱的量子行为的粒子群算法(QPSO)。

QPSO 算法的思想来源于量子力学和 PSO 模型。它能保证算法的全局收敛并且在优化模型中只有位置向量，没有速度向量，控制参数少，寻优能力强。尽管 QPSO 算法在优化问题上是一个有发展前景的算法，但是它像其他进化算法一样，也会遇到早熟收敛的问题，并且在收敛的后期粒子的多样性减少了。因此出现了许多改进的 QPSO 算法。Sun 等提出了概率分布机制使种群在全局搜索中更加有效；而且 Sun 等提出了多样性的 QPSO 来防止种群的聚集，使粒子更容易避开局部最优点；在 QPSO 算法中加入模拟退火(SA)不仅能跳出局部最优而且能够提高 QPSO 算法的全局搜索能力；Coelho 介绍了基于 Gaussian 概率分布的 QPSO 算法，在此算法中引入了变异算子；采用 Cauchy 变异来提高 QPSO 算法种群的多样性，加强算法的全局收敛能力；在 QPSO 算法中引入免疫算子，利用免疫记忆和接种技术的特征引导算法的搜索过程，来提高算法的收敛速度。

2.2　最优化理论

本课题利用群体智能算法进行最优化问题的优化设计。最优化问题是一个重要的数学分支，它所研究的问题是讨论在众多的方案中什么样的方案最优以及怎样找出最优方案。这类问题普遍存在，

例如，工程设计中怎样选择设计参数，使得设计方案既满足设计要求又能降低成本；在资源分配中，怎样分配有限资源，使得分配方案既能满足各方面的基本要求，又能获得好的经济效益；在生产计划安排中，选择怎样的计划方案才能提高产值和利润；在原料配比问题中，怎样确定各种成分的比例，才能提高质量，降低成本；在城建规划中，怎样安排工厂、机关、学校、商店、医院、住户和其他单位的合理布局，才能方便群众，有利于城市各行各业的发展；在农田规划中，怎样安排各种农作物的合理布局，才能保持高产稳产，发挥地区优势；在军事指挥中，怎样确定最佳作战方案，才能有效地消灭敌人，保存自己，有利于战争的胜利。在工程、技术、经济、管理和科学研究等众多领域中，最优化研究正是为这些问题的解决提供理论基础和求解方法，具有广泛的理论价值和应用价值。

最优化问题可以追溯到十分古老的极值问题，然而，直到 1947 年 Dantzig 提出求解一般线性规划问题的单纯形法之后，它才成为一门独立的学科。20 世纪 40 年代以来，由于生产和科学研究突飞猛进地发展，特别是电子计算机日益广泛应用，使最优化问题的研究不仅成为一种迫切需要，而且有了求解的有力工具。因此最优化理论和算法迅速发展起来，形成一个新的学科。至今已出现线性规划、整数规划、非线性规划、几何规划、动态规划、随机规划、网络流等许多分支。最优化理论和算法在实际应用中渗透到各个领域，并正在发挥越来越大的作用。

2.2.1 最优化问题

所谓最优化问题，就是指在满足一定的约束条件下，寻找一组参数值，以使某些最优性度量得到满足，即使系统的某些性能指标达到最大或最小。通常情况下，最优化问题是寻找最小值问题(寻找最大值问题可以转化为寻找最小值问题)。最优化问题根据其目标函数、约束函数的性质以及优化变量的取值等可以分成许多类

型，每一种类型的最优化问题根据其性质的不同都有其特定的求解方法。不失一般性，最小化问题可定义为

$$\begin{cases}\min \sigma = f(X) \\ \text{s.t. } X \in S = \{X \mid g_i(X) \leqslant 0,\ i = 1, \cdots, m\}\end{cases} \tag{2.9}$$

式中，$\sigma = f(X)$ 为目标函数；$g_i(X)$ 为约束函数；S 为约束域；X 为 n 维优化变量。

通常，对 $g_i(X) \geqslant 0$ 的约束和等式约束可以转化为 $-g_i(X) \leqslant 0$ 的约束。

当 $f(X)$、$g_i(X)$为线性函数，且 $X \geqslant 0$ 时，上述最优化问题即为线性规划问题，其求解方法有成熟的单纯形法和 Karmarc 方法。

当 $f(X)$、$g_i(X)$中至少有一个函数为非线性函数时，上述问题即为非线性规划问题。非线性规划问题相当复杂，其求解方法多种多样，但到目前仍然没有一种有效地适应所有问题的方法。

当优化变量 X 仅取整数值时，上述问题即为整数规划问题，特别是当 X 仅能取 0 或 1 时，上述问题即为 0-1 整数规划问题。由于整数规划问题属于组合优化范畴，其计算量随变量维数的增长而指数增长，所以存在着“维数灾难”问题。

当 $g_i(X) \leqslant 0$ $(i=1,\cdots,m)$所限制的约束空间为整个 n 维欧氏空间，即 $\mathbf{R}^n$ 时，上述最优化问题为无约束优化问题，即

$$\begin{cases}\min \sigma = f(X) \\ \text{s.t. } X \in S \subset \mathbf{R}^n\end{cases} \tag{2.10}$$

非线性规划问题(包括无约束优化问题和约束优化问题)，由于函数的非线性，使得问题的求解变得十分困难，特别是当目标函数在约束域内存在多峰值时。常见的求解非线性问题的优化方法，其求解结果与初值的选择关系很大，也就是说，一般的约束或无约束非线性优化方法均是求目标函数在约束域内的近似极值点，而非真正的最小点。近几年来，随着计算机技术的发展，一些过去无法解决的复杂优化问题，已经能够通过计算机来求得近似解，所以计算

机求解优化问题的方法研究就显得越来越重要。计算机求解优化问题的主要手段就是对优化问题的可行解空间进行搜索，而按照搜索策略的不同，可以将主要的搜索方法分为三类：

(1) 枚举法。在可行解空间内枚举出所有可行解，以求出精确最优解。对于连续问题，该方法要求先对其进行离散化处理，这样就有可能产生离散误差而永远达不到最优解。另外，当枚举空间比较大时，该方法的求解效率比较低。枚举法的策略最简单，计算量也最大。而且枚举法只能应用于可行解空间是有限集合的情形。

(2) 启发式算法。寻求一种能产生可行解的启发式规则，以找到一个最优解或近似最优解。该方法的求解效率虽然比较高，但对每一个需要求解的问题都必须找出其特有的启发式规则，这种启发式规则无通用性，不适合于其他问题。

(3) 搜索算法。该算法在可行解空间的一个子空间内进行搜索操作，以找到问题的最优解或近似最优解。该方法虽然不一定保证能够得到问题的最优解，但若适当地利用一些启发知识，就可以较好地平衡近似解的质量和求解效率。

2.2.2 局部优化算法

如果存在 $X_D^* \in D$，使得对于 $\forall X \in D$ 有

$$f(X_D^*) \leqslant f(X), X \in D \tag{2.11}$$

成立，其中 $D \subset S \subseteq \mathbf{R}^n$，$S$ 为由约束函数限定的搜索空间，则称 X_D^* 为 $f(X)$在 D 内的局部极小点，$f(X)$为局部极小值。

常见的优化方法大多为局部优化方法，都是从一个给定的初始点 $X_0 \in S$ 开始，按照某种方法寻找下一个使得目标函数值更小的解，直至满足某种停止准则。成熟的局部优化方法很多，如Newton-Raphson 法、共轭梯度法、Polar-Ribiere 法、Davidon-Fletcher-Power(DFP)法、Broyden-Fletcher-Goldfarb-Shsnn(BFGS)方法等，还有专门为求解最小二乘问题而发展的 Levenberg-Marquardt(LM)算

法。所有这些局部优化算法都是针对无约束优化问题的，而且对目标函数均有一定的解析性质要求，如 Newton-Raphson 法要求目标函数连续可微，同时要求其一阶导数连续。

2.2.3　全局优化算法

全局最优化问题通常可描述为：令 S 为 $\mathbf{R}^n$ 上的有界子集(即变量的定义域)，f: $S \to \mathbf{R}$ 为 n 维实值函数，所谓函数 f 在 S 域上全局最小化就是寻求点 $X_{\min} \in S$ 使得 $f(X_{\min})$在 S 域上全局最小，即

$$\forall X \in S: f(X_{\min}) \leqslant f(X) \tag{2.12}$$

到目前为止，全局优化问题也已存在了许多算法，如填充函数法等，但比起局部优化问题的众多成熟方法，其间还有很大差距。另外，解析性优化方法对目标函数及约束域均有较强的解析性要求，对于诸如目标函数不连续、约束域不连通、目标函数难以用解析函数表达或者难以精确估计等问题，解析确定性优化方法就难以适应。

为了可靠解决全局优化问题，人们试图离开解析确定型的优化算法研究，转而探讨对函数解析性质要求较低甚至不做要求的随机型优化方法。真正有效且具有普遍适应性的随机全局优化方法，是近几十年来人们模拟自然界的一些自然现象而发展起来的一系列仿生型智能优化算法，如禁忌搜索算法、模拟退火算法、进化类算法、群体智能算法等。

2.2.4　最优化问题的求解

1. 传统方法求解

以成熟的最速梯度法和牛顿法等为代表的传统优化算法具有完善的数学基础、计算效率高和可靠性强等特点，是一类最重要、应用最广泛的优化算法。这类算法的基本迭代步骤如下：

(1) 给定初始点 $x(0)$，i=0。

(2) 按照某一方法或规则构造搜索方向 $d(i)$。

(3) 确定步长 l_i。

(4) 计算下一个迭代点 $x(i+1)=x(i)+l_i d(i)x$。

(5) 判断 $x(i+1)$是否满足终止条件，若满足，则停止迭代，$x(i+1)$是局部近似最优解，求解结束；否则，$i=i+1$，转(2)。

在迭代中，核心是构造搜索方向 $d(i)$和确定步长 l_i。总体来看，传统的基于梯度的优化算法普遍要求目标函数导数连续、具有计算复杂、串行求解等特点。同时，在面对离散、不连续、无导数、高度病态的优化问题时，它们无能为力。此外，传统的优化算法建立在局部下降的基础上，常常无法求得全局最优解。

2. 智能算法求解

近三十多年来，以遗传算法为代表的进化算法(EA)作为智能算法中重要的分支，为求解最优化问题提供了新的思路和方法。EA 模拟了生物进化过程和机制来求解问题，即认为生物进化是一个从简单到复杂、从低级到高级，自然、并行发生且稳健的优化过程，遵循“物竞天择，适者生存”的法则。进化算法作为一种基于种群的随机优化算法，其计算过程可描述为：

(1) 借助一定的问题信息或者随机生成一组初始化解，作为初始种群。

(2) 对当前种群的个体进行评价。

(3) 检验当前种群的个体是否满足进化结束条件，若满足，则算法终止，输出最优解和最优值。

(4) 依据一定的规则从当前种群中选择个体，构成新的种群，开始下一代进化。

(5) 对新的种群施加进化算子，产生子代个体，转(2)。

与传统的基于梯度的优化算法相比，进化算法具有以下特点：

(1) 进化算法具有自组织、自适应和自学习能力。在确定了适应度函数和进化算子后，进化算法将利用进化过程中获得的信息自

行组织搜索，使得适应度大的个体具有较高的生存概率。

(2) 进化算法不需要导数或其他辅助知识，而只需要影响搜索方向的目标函数或相应的适应度函数。

(3) 进化算法具有本质并行性。进化算法具有内在并行性，即算法本身适合大规模并行；同时具有内含并行性，由于进化算法采用种群的方式组织搜索，因而可以同时搜索解空间内的多个区域，并相互交流。

(4) 进化算法是随机搜索算法，强调概率转换规则，而不是确定的转换规则，是一种全局优化算法。

(5) 对一个给定的问题，进化算法可以同时产生多个潜在解，最终由使用者根据需要选择使用。

始于 20 世纪 90 年代研究的群体智能算法是进化算法的一个重要发展。其基本思想是模拟自然界生物群体行为来构造随机优化算法。群体智能算法对函数形态要求较弱、寻优结果和初值无关，并具有一定的并行性，因而已成为学术界研究的一个热点。

2.3　本章小结

本章简要介绍了六种进化算法的概念和它们的优化过程，以及算法的改进。关于遗传算法、粒子群优化算法和量子粒子群优化算法的详细介绍见第 3 章。进化算法的主要功能是对问题进行优化，因此本章介绍了最优化理论以及最优化理论的求解。

参 考 文 献

[1] Kennedy J, Eberhart R C. Particle Swarm Optimization [C]. In: Proceedings of IEEE International Conference on Neural Networks.

1995: 1942-1948.

[2] Sun J, Feng B，and Xu W B. Particle Swarm Optimization with Particles Having Quantum Behavior[C]. In: Proceedings of IEEE 2004 Congress on Evolutionary Computation. USA: Portland, 2004: 325-331.

[3] Koza J R. Genetic Programming[M]. MIT Press, 1991.

[4] Rechenberg I. Evolutions strategie: Optimierung Technischer Systeme Nach Prinzipien der Biologischen Evolution[M]. Stuttgart: Frommann-Holzboog, 1973.

[5] Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs[M]. 2nd ed. Springer-Verlag, 1994.

[6] Kennedy J, Eberhart R C. A Discrete Version of the Particle Swarm Algorithm[C]. In: Proceedings of the 1997 Conference on System, Man and Cybernetics, Piscataway. NJ: IEEE Service Center, 1997: 4104-4109.

[7] Shi Y, Eberhart R C. A Modified Particle Swarm Optimizer[C]. In: Proceedings of the IEEE International Conference on Evolutionary Computation, Piscataway, NJ: IEEE Press, 1998: 69-73.

[8] Yuhui S, Eberhart R C. Fuzzy Adaptive Particle Swarm Optimization[C]. In: Proceedings of the 2001 Congress on Evolutionary Computation, 2001: 101-106.

[9] Clerc M. The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization[C]. In: Proceedings of the 1999 Congress on Evolutionary Computation. 1999: 1953-1957.

[10] Angeline P J. Using Selection to Improve Particle Swarm Optimization[C]. In: The 1998 IEEE International Conference on Evolutionary Computation. 1998: 84-89.

[11] Suganthan P N. Particle swarm optimiser with neighbourhood operator[C]. In: Proceedings of the 1999 Congress on Evolutionary Computation. Piscataway, NJ, USA: IEEE Press, 1999: 1958-1962.

[12] Kennedy J. Small Worlds and Mega-minds: Effects of Neighborhood Topology on Particle Swarm performance[C]. In: Proceedings of the 1999 Congress on Evolutionary Computation. 1999: 1933-1938.

[13] Lovbjerg M, Rasussen T K, Krink T. Hybrid Particle Swarm Optimiser with Breeding and Subpopulations[C]. In: Proceedings of the third Genetic and Evolutionary Computation Conferences. San Francisco, 2001: 469-476.

[14] Kennedy J. Probability and Dynamics in the Particle Swarm[C]. In: 2004 Congress on Evolutionary Computation. 2004: 340-347.

[15] Krohling R A. Gaussian Swarm: a Novel Particle Swarm Optimization Algorithm[C]. In: 2004 IEEE Conference on Cybernetics and Intelligent Systems. 2004: 372-376.

[16] Riget J, Vesterstroem J. A Diversity-Duided Particle Swarm Optimizer — the ARPSO[R]: Department of Computer Science, University of Aarhus, 2002.

[17] 崔志华，曾建潮. 基于控制理论的微粒群算法分析与改进[J]. 小型微型计算机系统，2006(05)：849-853.

[18] 崔志华，曾建潮. 基于微分模型的改进微粒群算法[J]. 计算机研究与发展，2006(04)：646-653.

[19] 王丽芳，曾建潮. 基于微粒群算法与模拟退火算法的协同进化方法[J]. 自动化学报，2006(04)：630-635.

[20] 高海兵，周驰，高亮. 广义粒子群优化模型[J]. 计算机学报，2005(12)：1980-1987.

[21] 窦全胜，周春光，马铭. 粒子群优化的两种改进策略[J]. 计算机研究与发展，2005(05)：897-904.

[22] 刘宇，覃征，史哲文. 简约粒子群优化算法[J]. 西安交通大学学报，2006(08)：883-887.

[23] 刘洪波，王秀坤，谭国真. 粒子群优化算法的收敛性分析及其混沌改进算法[J]. 控制与决策，2006(06)：636-645.

[24] 俞欢军，张丽平，陈德钊，等. 基于反馈策略的自适应粒子群优化算法[J]. 浙江大学学报(工学版)，2005(09)：12-17.

[25] Clerc M, Kennedy J. The Particle Swarm: Explosion, Stability and Convergence in a Multi-Dimensional Complex Space [J]. IEEE Transactions on Evolutionary Computation. 2002(6): 58-73.

[26] Sun J, Xu W B, Fang W. Quantum-Behaved Particle SwarmOptimization with a Hybrid Probability Distribution[C]. In the proceeding of 9th Pacific Rim International Conference on Artificial Intelligence. PRICAI, 2006: 737-746.

[27] Sun J, Xu W B, Fang W. Enhancing Global Search Ability of Quantum-BehavedParticle Swarm Optimization by Maintaining Diversity of the Swarm[C]. In: RSCTC 2006, 2006: 736–745.

[28] Liu J, Sun J, Xu W B. Improving quantum-behaved particle swarm optimization by simulated annealing[C]. In: LNBI 4115. Heidelberg: Springer, 2006: 130–136.

[29] Coelho L S. Novel Gaussian Quantum-Behaved Particle Swarm Optimizer Applied to Electromagnetic Design[J]. Science Measurement & Technology, 2007, 11(5): 290-294.

[30] Liu J, Sun J, Xu W B. Quantum-behaved particle swarm optimization with mutation operator[C]. In: Proc. 17th IEEE International Conference on Tools with Artificial Intelligence. HK, 2005: 237-240.

[31] Liu J, Sun J, Xu W B. Quantum-Behaved Particle Swarm Optimization with Immune Operator[C]. In: Proc 16th International Symposium on Methodologies for Intelligent. Italy: Springer, 2006: 77-83.

第 3 章　遗传算法、粒子群优化算法和量子粒子群优化算法

3.1　遗传算法

3.1.1　遗传算法的基本思想

遗传算法(GA)的概念最早是 1967 年由 Bagley 和 Rosenberg 提出的，1975 年美国 Michigan 大学的 Holland J H 教授等在著作 *Adaptation in Natural and Artificial Systems* 又再次提到遗传算法的概念，他们将达尔文的进化论和孟德尔的遗传学说引入到各种实际的工程问题中，并对其进行优化，这些研究成为遗传算法的基础。遗传算法是以自然界中的“适者生存”的生物进化理论为背景，模拟自然选择和遗传进化过程中的繁殖、交配和变异等现象，进而发展起来的一种高效随机搜索优化方法。

遗传算法的基本原理是从任意一个初始种群出发，应用群体搜索技术，将种群代表一组问题解，通过对当前种群的选择、交叉和变异等遗传操作，产生新一代种群，并逐步使种群进化到包含近似最优解的状态。其主要特点是群体搜索策略和群体中个体之间的信息交换，在搜索过程中不依赖于外界信息，而是有效利用已有信息来自动获取和积累有关搜索空间的知识，并自适应地控制搜索方向使其最终找到最优解。遗传算法与进化策略、进化规划共同构成了进化算法的主要框架。

3.1.2　遗传算法中的基本术语

为了能更好地理解遗传算法的思想，下面介绍遗传算法中涉及的相关术语。

- 串(string)：个体的表现形式，对应于遗传学中的染色体(chromosome)，即生物细胞中含有的一种遗传物质基因的载体。
- 基因(gene)：串中部分字符的组合元素，表示不同的特征。对应于遗传的基本单位，以碱基序列表现。
- 基因座位(locus)：某一遗传基因在染色体中的位置。在遗传算法中，可以将其看作序列的位置。
- 个体(individual)：带有特征的染色体实体，是遗传算法中所处理的基本对象和结构，可看作实际问题的解。
- 种群(population)：个体的集合，集合中个体的数目称为种群的规模或大小。
- 选择(selection)：在有限资源空间的排他性竞争，以一定概率从种群中选择若干个体的过程。在遗传算法中通过选择获得适应度较高的个体。
- 交叉(crossover)：染色体间通过交叉使基因重组，又称为杂交。在遗传算法中按照一定规则交换部分基因，是遗传算法中的主要操作，也是产生新个体的主要方法。
- 变异(mutation)：染色体基因被复制时，可能以小概率产生某些复制差错，从而产生新的染色体。在遗传算法中将个体编码串中的某些基因值用其他基因值替换，从而形成一个新个体的操作方式。
- 适应度(fitness)：是用来度量某物种在遗传和进化中对其生存环境的适应程度，或在环境压力下的生存能力，取决于遗传特性。在遗传算法中表示为适应度函数，其函数值越高，说明个体的质量越好。

- 编码(coding)：是通过某种编码机制将实际问题解空间的解数据表示成串形式的数据结构，在遗传算法中表现型到遗传型的映射。
- 解码(decoding)：遗传操作结束后，遗传型到表现型的映射。

3.1.3　遗传算法的步骤及流程图

遗传算法是通过对进化过程中的种群反复进行选择、交叉、变异操作来模拟自然界中种群的演变过程，直到满足一定性能要求才结束计算。应用遗传算法求解问题的基本步骤如下：

(1) 编码。确定变量定义域及编码精度，形成编码方案。

(2) 种群初始化。随机产生初始种群作为第一代，同时确定个体长度和种群规模。

(3) 计算适应度值。根据问题设置合适的适应度函数，计算种群中个体的适应度，并判断是否满足终止条件，若满足，则输出最佳个体及其代表的最优解并结束计算；否则进入下一步。

(4) 遗传操作之选择算子。依据适应度值从当前种群中选择优良的个体，使它们有机会被选中进入下一代种群中，适应度高的个体被选中的概率高，适应度低的个体可能被淘汰，体现了进化论的“适者生存”原则。

(5) 遗传操作之交叉算子。对被选择进入匹配池中的父辈个体进行交叉操作，形成子辈个体，得到新种群，体现了信息交换的原则。

(6) 遗传操作之变异算子。以小概率在种群中随机选择某个父辈个体进行变异操作，形成子辈个体，得到新种群。

(7) 更新种群。根据一定概率选择个体形成新种群，返回到步骤(3)。

其基本流程如图 3.1 所示。

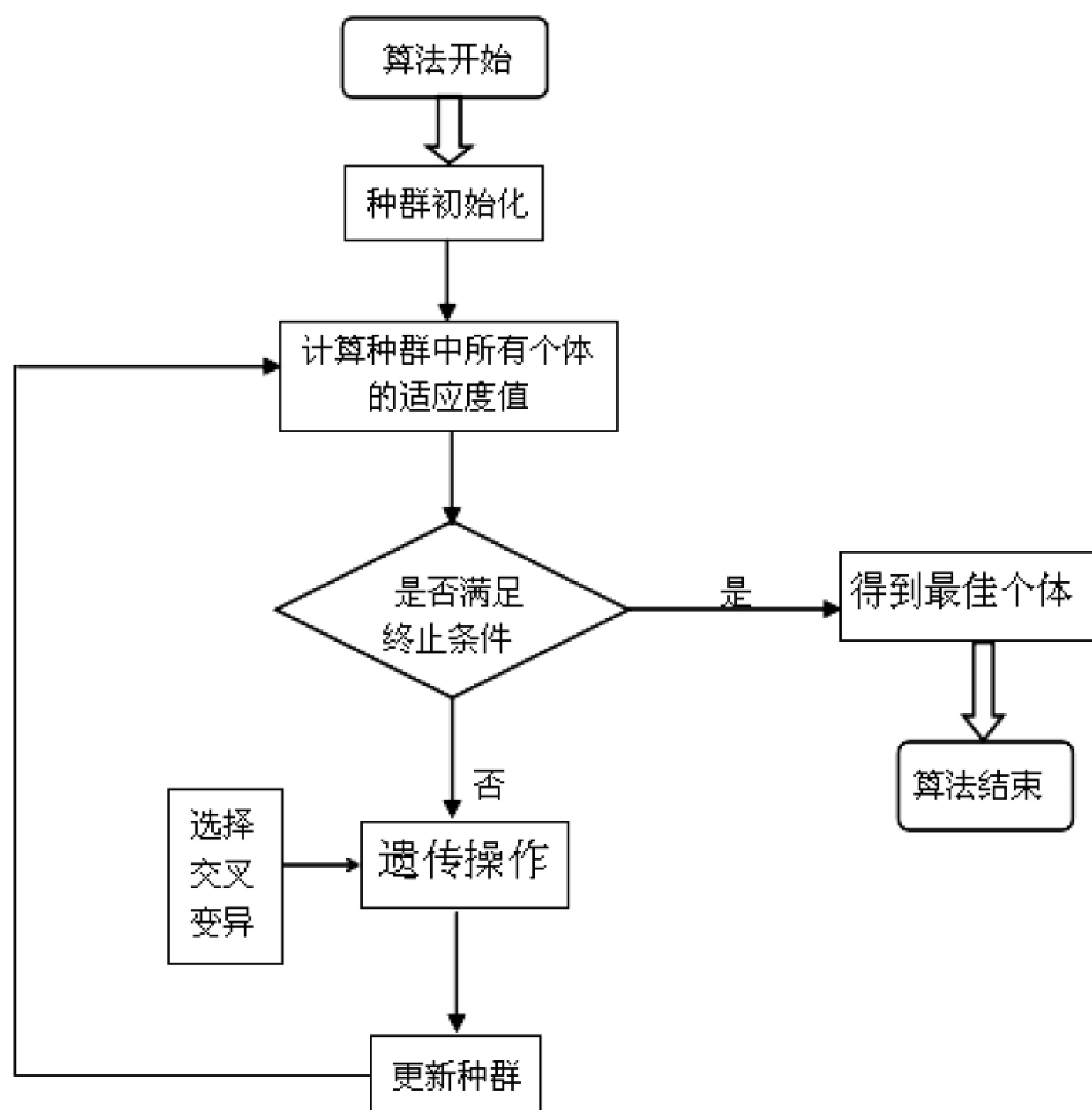


图 3.1　遗传算法基本流程图

3.1.4　遗传算法的构成要素

根据上面的基本步骤和流程图可以看出，遗传算法中有五个关键的构成要素。

1. 编码

编码就是用一种码来表示优化问题的解，是从实际问题到解空间数学模型的映射。因此对于一个实际的优化问题，首先要将其表示为适合于遗传算法操作的形式，即编码。所谓编码，就是把问题的搜索空间中每个可能的点表示成遗传算法可以处理的格式，解码则与之相反。当算法找到最佳个体后，要进行相应的解码操作后，才能够得到

实际问题的解。随着编码方案的确定，解码方案也就随之确定。

编码是应用遗传算法解决实际问题的首要问题，也是关键问题。它决定了个体中基因的排列次序，也决定了如何进行交叉、变异等操作。编码方法的好坏，不仅影响了遗传操作实现的难易程度，而且在很大程度上决定了遗传算法的执行效率。Balakrishman 等较全面地讨论了编码的一组特性，主要包括可扩展性、完全性和复杂性等九个特性，但设计编码策略时仍需要在这些彼此矛盾特性中权衡利弊。

常用的编码方式有以下几种。

- 二进制编码：是遗传算法编码中最常用的方式，其编码符号集是二进制的 0 和 1 字符串，其个体基因是二值符号串。例如(0110011)就是一个长度为 7 的二进制编码个体。本书第 5 章粒子群优化算法的编码就是采用这种编码方式。
- 十进制编码：其编码符号集是十进制的 0～9 的字符串，其个体基因是十值符号串。例如(0234761)就是一个长度为 7 的二进制编码个体。
- 符号编码：其编码符号集是无数值意义只有代码意义的符号集，这个符号集可以是一个字母表，如{A,B,C,…}，也可以是一个序号表{1,2,3,…}。本书第 4 章遗传算法的编码就是采用这种字母符号编码方式。
- 浮点数编码：个体的每个基因值都是一个浮点数，一般是决策变量的真实值。该方法适用于在遗传算法中表示较大的数，应用于高精度的遗传算法。

2. 产生初始种群

初始种群是遗传算法进行搜索最优解的开始，是由多个染色体(个体)组成的集合。产生初始种群的方法一般有两种：一种是完全随机地产生初始种群，适用于对待求解问题的解没有任何先验知识的情况，这样产生的种群具有更好的多样性，在一定程度上可以避免收敛于局部最优解；另一种是根据某些已知的先验知识转化为必

须满足的一组要求，在初始化时对要生成的解加以限定，然后在满足这些要求的解中进行随机选取，这样产生的种群运行时能够更快地收敛到最优解。

产生初始种群的关键是种群规模。规模越大，被遗传操作处理的模式越多，搜索到问题最优解的机会就越高，算法也不容易陷入局部最优解。反之，如果规模较小，则利用适应度函数进行评价的次数就少，不但减少了计算量，而且对系统资源的要求也降低了，从而提高了算法的执行效率；但是规模太小，又降低了种群的多样性，影响算法的全局优化性能。因此，群体规模太大或太小与算法更快、更精确地得到最优解有着直接的影响。在实际应用中，群体规模一般取值范围为 20～100。

3. 适应度函数

在用遗传算法进行进化搜索最优解之前，必须要确定适应度函数。适应度函数是用来评价种群中个体优劣程度的一个指标。适应度函数体现了自然界进化过程中，各种生物对自然环境的适应能力，适应度值越大繁衍能力越强；值越小则越可能灭亡。

一般而言，遗传算法在搜索过程中，仅以适应度函数为依据，利用种群中个体的适应度值来决定搜索的方向，而基本不用外部信息。适应度函数的选取十分重要，是决定遗传算法收敛速度以及能否得到全局最优解的关键点。如果适应度函数设计不当，则可能在遗传算法后期出现收敛于局部最优解的情况。因此，适应度函数是影响遗传算法收敛速度以及能否找到最优解的一个重要因素。

一般情况下适应度函数由目标函数或费用函数转变而成。适应度函数的设计应确保适应度函数单值、非负、连续，并尽量满足设计简单、通用性强的特点。

4. 遗传操作

遗传操作主要包括选择、交叉和变异三种基本操作。

1) 选择操作(Selection)

选择操作是按一定规则从父代种群中选取若干个体遗传到下一代种群中的操作方式。在遗传算法中，适应度高的个体被遗传到下一代种群中的概率也大，这样可以加强优秀个体在下一代中的优势。群体中某个个体被选择的概率与其适应度值成正比。常用的选择算子包括最佳个体保存选择、期望值选择、轮盘赌选择、随机遍历抽样选择和截断选择等。下面详细讲述三种最常用的选择算子。

(1) 比例选择(或轮盘赌选择)

比例选择方法也叫轮盘赌选择方法，在这种选择方法中，各个个体被选中的概率与其适应度成正比，个体的适应度越高，其被选中的概率就越大。由于该方法思想简单且容易实现，因此它是遗传算法中最经常使用的选择方法。轮盘赌选择的基本思想是：根据个体 i 的适应度值 $f_i (i=1,\cdots,k)$，计算出个体的相对适应值 $f_i \Big/ \sum_{j=1}^{k} f_j$，记为 p_i，然后根据选择概率 $\{p_i, i=1,\cdots,k\}$ 把一个圆盘分成 k 份，其中第 i 个扇形的圆心角为 $2\pi p_i$。在进行选择时可以想象一下转动圆盘，若某个参照点落入第 i 个扇形中，则选择个体 i。

实现过程为：首先生成一个[0,1]的随机数 r，若 $p_1 + p_2 + \cdots + p_{i-1} < r < p_1 + p_2 + \cdots + p_{i-1} + p_i$，则选择个体 i。可见，适应度值高的个体被选择的概率也大，其基因更容易被遗传到下一代。

缺点：随机操作原因，误差比较大，有时适应度高的个体会被淘汰。

(2) 最佳个体保存策略(或精英保留法)

在使用遗传算法求解问题的过程中，虽然随着群体的进化过程会产生出越来越多的优良个体，但由于选择、交叉、变异等遗传操作的随机性，当前群体中适应度最好的个体也有可能被破坏掉，从而降低了群体的平均适应度，影响遗传算法的运行效率和收敛速度。为此，还经常使用将适应度最好的个体保留到下一代群体中的方法来进行优胜劣汰操作，即当前群体中适应度最高的个体不参与交叉运算和变

异运算，而是用它来替换掉本代群体中经过交叉、变异等遗传操作后所产生的适应度最低的个体。最佳个体保存法的思想是把群体中适应度最高的个体不进行配对交叉，而直接复制到下一代中。这种选择操作又称为复制。通过复制，每代最优个体的基因不会被交叉、变异等操作破坏，使算法向最优解迫近。这里可以将上一代个体的适应度值按大小排列，将前 10%的个体保存到下一代。具体操作过程如下：

① 找出当前群体中适应度最高和最低的个体。

② 若当前群体中最佳个体的适应度比总的迄今为止的最好个体的适应度高，则以当前群体中的最佳个体作为新的迄今为止的最好个体。

③ 用迄今为止的最好个体替换当前群体中的最差个体。

(3) 确定式采样选择

使用以上两种方法进行个体选择时，选择操作的随机性很强，不依赖于人的意志而改变。确定式采样选择方法可以人为地控制对个体的选择操作，其基本思想是按照一种确定的方式来进行选择。具体操作过程如下：

① 计算群体中每个个体在下一代中的生存期望数目 $N_i = M - F_i \Big/ \sum_{i=1}^{M} F_i \quad (i = 1, 2, \cdots, M)$。

② 用 N_i 的整数部分确定各个对应个体在下一代群体中的生存数目。由该步可确定出下一代群体的 $\sum_{i=1}^{M} \{N_i\}$，其中 M 为群体中个体的数目。

③ 按照 N_i 的小数部分对个体进行降序排序，顺序取前 $M - \sum_{i=1}^{M} [N_i]$ 个个体加入下一代群体中。

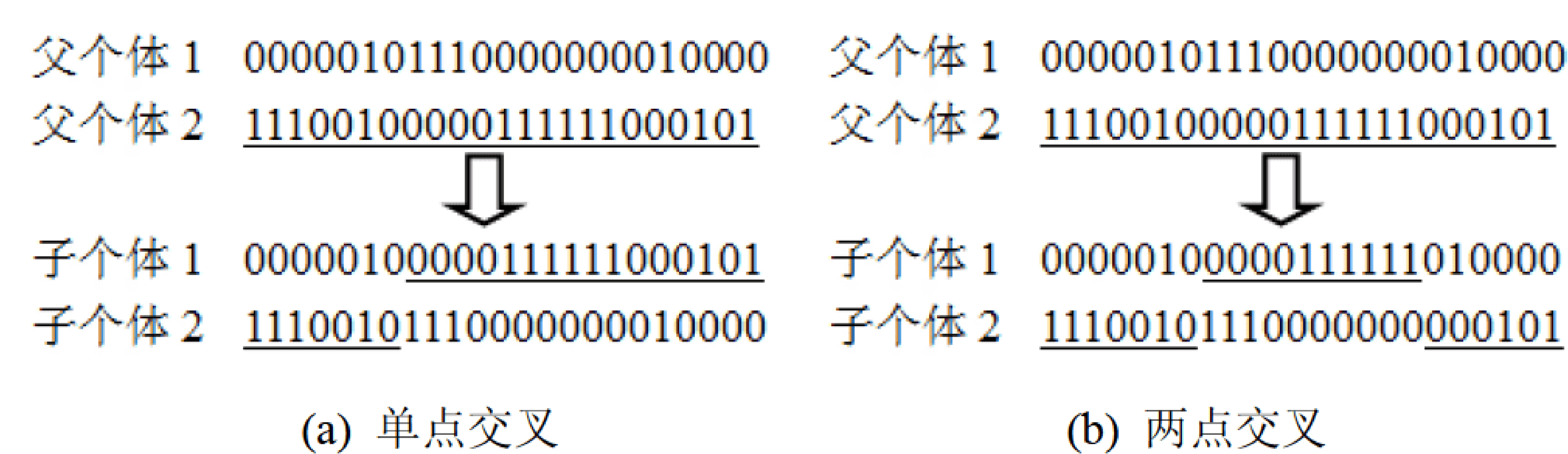
④ 下一代中的 M 个个体全部确定出来。

优点：能够保证适应度较大的一些个体一定能够被保留到下一代群体中。

2) 交叉操作(Crossover)

交叉操作是将两个父代个体按一定规则相互交换部分基因，从而形成两个新的个体的操作方式。交叉操作是遗传算法中最主要的操作，是产生新个体的主要操作方法，决定了遗传算法的全局搜索能力。交叉算子将被选中的两个个体的基因按交叉概率进行交叉，产生两个新的个体，被交换基因的位置是随机的。交叉概率的大小，决定了交叉操作的频率。频率越大，可以越快地收敛到最可能包括最优解的区域；但频率太高也可能会导致算法过早收敛。实际应用中，交叉概率一般取值在 0.4～0.9 之间。

交叉操作的方式有单点交叉、多点交叉等。图 3.2 所示是两种常见交叉操作的示意图。



(a) 单点交叉　　(b) 两点交叉

图 3.2　单点交叉和两点交叉

单点交叉(one-point crossover)是在个体中随机设定一个交叉点，交叉时该点前或后的两个个体的部分结构进行互换，生成两个新的个体。图 3.2(a)举例说明了多序列比对的一点交叉法。在两个父个体中随机选取父比对 1 中第一条序列第七个位置为交叉位置，从此处垂直切开，将父比对 2 从与父比对 1 相应的字符位置处切开，然后将切开后的两个序列的前后部分交换，互换拼接后的两部分之间加空格补齐，使每个序列个体的长度相等，得到两个新的子个体。所产生的两个个体，与之前的父代个体比较，只保留适应度值高于父代的新个体。交叉操作是按概率 p_c 来控制发生的，并不是所有被选择到的个体都会发生交叉，这样的设计符合生物进化的规律。

3) 变异操作

变异操作是根据变异概率将个体编码串中的某些基因值用其他基因值替换从而形成一个新个体的操作方式。变异是遗传算法中产生新个体的辅助方法。在变异操作中，虽然增大变异概率可以保证种群的多样性，但过大会导致遗传算法退化为“纯”随机搜索算法。变异概率的大小一般与染色体的长度成反比，与种群的大小无关。实际应用中，变异概率一般取值范围为 0.001～0.1。

变异操作能够提高遗传算法的局部搜索能力，保持种群多样性，使算法免于陷入局部最优解而停滞不前。变异操作包括实值变异和二进制变异两种，其中二进制变异方法包括换位、复制、插入、删除等。如果按照变异位点个数来分类，又分为一点变异和多点变异。

图 3.3 所示是常见的一点变异操作的示意图。

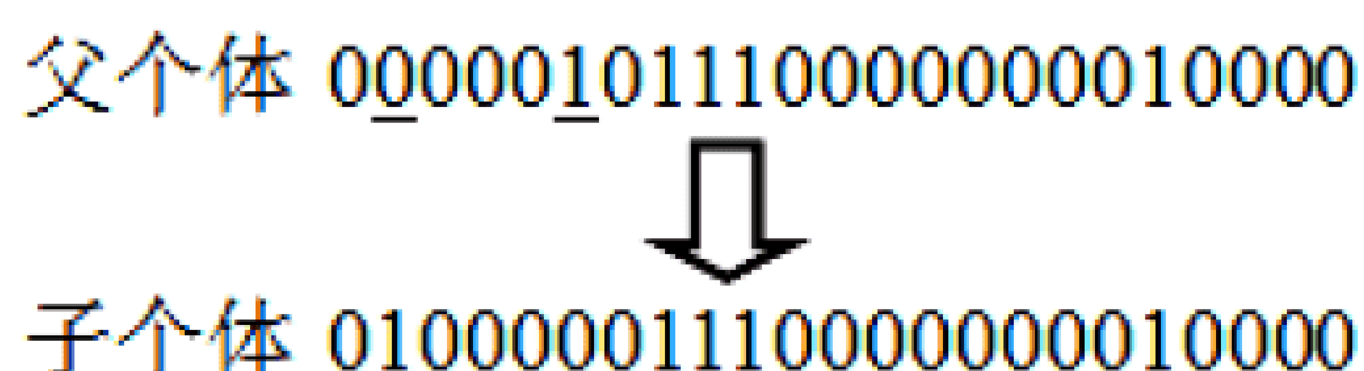


图 3.3　一点变异

5. 终止条件及参数设定

遗传算法的终止条件一般有以下几种：

(1) 种群中个体的最大适应度值超过预设值。

(2) 种群中个体的平均适应度值超过预设值。

(3) 种群中最大世代数超过预设值。

(4) 遗传操作失败次数超过预设值。

遗传算法运行参数包括染色体长度、种群大小、交叉率、变异率、最大迭代数等。这些参数一般可在算法运行前根据实际情况自行设定。

3.1.5　遗传算法的优缺点

遗传算法不同于传统的搜索和优化方法，它主要具有以下几方

面优点：

(1) 具有自适应、自组织和自学习性(智能性)。遗传算法根据进化过程获得的信息可以自行调整搜索方向，对于适应度大的个体将具有较高的生存概率，而适应度低的个体将在进化过程中消亡。

(2) 遗传算法具有本质并行性。许多传统搜索算法都是进行单点搜索，因此极易陷入局部的最优解。而遗传算法是从多个初始个体一齐进行搜索，能够有效地进行全局搜索，减少了陷入局部最优解的风险，同时提高了计算速度。

(3) 遗传算法应用范围广泛且应用灵活。利用遗传算法解决问题时基本上不用其他辅助信息，只需要设计合理的影响搜索方向的适应度函数，且对函数本身基本无限制。这一特点使得遗传算法的应用范围大大扩展。

(4) 遗传算法不是对参数本身进行操作，而是对把参数集进行了编码的个体进行操作。这一特点使得遗传算法应用更加直接、方便。

(5) 遗传算法不是盲目搜索。它采用概率的变迁规则来指导其搜索方向，而随机性操作有效保持了种群的多样性。

遗传算法以其鲁棒性强、应用灵活等优点，在大多数情况下都可以得到最优解，对大规模 NP 问题也能取得比较理想的效果。

缺点：遗传算法虽然在很多方面已取得成功的应用，但在各方面的应用中普遍存在过早收敛于局部最优解的现象，即算法的“早熟”问题，以及在进化后期搜索效率低。遗传算法是一种全局搜索算法，但是仍然存在无法收敛于全局最优解的问题。最典型的是“早熟”收敛现象。“早熟”是指在遗传算法迭代过程中种群多样性遭到破坏，算法搜索结果停滞不前，最终收敛于一个局部最优解，而无法得到全局最优解。由于遗传种群规模有限，传统遗传操作的按适应度比例选择、交叉、变异机制，以及对新产生的适应度较低的个体立即抛弃的策略，使得高于种群平均适应度的个体在下一代中得到较大的生存机会，这样不断地迭代进行，一旦某些个体取样在

种群中占优势，传统遗传算法就会强化这种优势，从而使得搜索范围迅速变窄，产生“近亲繁殖”，大大影响全局最优解的搜索。

3.1.6　遗传算法的应用现状

遗传算法提供了一种求解复杂系统优化问题的通用框架，它不依赖于求解问题的具体领域，且对问题的种类有很强的鲁棒性，所以被广泛应用于很多领域。下面是遗传算法的一些主要应用领域。

1. 函数优化

函数优化是遗传算法的经典应用领域，也是对遗传算法进行性能评价的常用算例。尤其是对于一些非线性、多模型、多目标的函数优化问题，用其他优化方法较难求解，而通过遗传算法却可以方便地得到较好的结论。

2. 组合优化

对于复杂的组合优化问题，在现有条件下有时很难甚至不可能得到精确最优解，因此，找到满意解成为了解决该类问题的有效手段。遗传算法是寻求这种满意解的最佳工具之一，并且对组合优化中的 NP 完全问题非常有效。

3. 生产调度问题

生产调度问题在许多情况下建立起来的数学模型都难以求得精确解，遗传算法是解决该类问题的有效工具。例如，在流水线生产车间调度、生产规划、任务分配等方面的有效应用。

4. 自动控制

遗传算法在自动控制领域解决优化问题方面显示了良好的效果，例如基于遗传算法的模糊控制器优化设计、利用遗传算法进行

人工网络的结构优化设计和权值学习等。

5. 机器人智能控制

机器人是一类复杂的难以精确建模的人工系统，而遗传算法的起源就来自于对人工自适应系统的研究，因此机器人智能控制是遗传算法的一个重要应用领域。

6. 图像处理和模式识别

图像处理和模式识别是计算机视觉中的一个重要研究领域，如何使图像处理过程中的误差最小是遗传算法在图像处理中进行优化计算的主要目的。目前遗传算法在图像恢复、图像边缘特征提取、几何形状识别等方面得到了应用。

7. 人工生命

人工生命与遗传算法有着密切的关系，基于遗传算法的进化模型是研究人工生命现象的重要理论基础。虽然人工生命的研究尚处于启蒙阶段，但遗传算法已显示了其卓越的应用能力。可以预见，在未来的研究发展中，遗传算法在人工生命领域中的应用将得到更为深入的发展。

8. 遗传程序设计

遗传程序设计是研究对所进行的遗传操作自动生成计算机程序的领域，它与遗传算法的基本思想相似，是遗传算法应用的领域之一。

9. 机器学习

基于遗传算法的机器学习，特别是分类器系统，在许多领域中都得到了应用。例如，基于遗传算法的机器学习可用于调整人工神经网络的连接权，也可以用于神经网络结构的优化设计等。

3.1.7　遗传算法的改进

基本遗传算法具有自组织性、自适应性、并行性、不确定性等特点，但是它也存在一些不足，如遗传算法的局部搜索能力差，容易出现早熟现象，使得收敛性能下降，极大影响了算法的搜索效率。因此众多学者一直致力于改进遗传算法，提高算法的计算效率。通常的改进思路有以下两种。

1. 基本结构的改进

主要是对遗传算法的基本结构进行改进，如编码方式、控制参数、初始种群、选择操作、交叉操作、变异操作等进行了深入的探究；或者针对适应值函数引入了动态策略和自适应策略以改善遗传算法的性能等。Cedric Notredame(1996)在遗传算法中对 22 种交叉变异算子应用了一个自动调度机制；刘立芳和霍红卫(2006)又提出六种新的遗传算子；Fan(2012)提出智能算子在遗传算法中的应用；司徒浩臻(2006)提出择优使用算子和优化算子组合的策略；C. Goondro(2007)认为初始化种群的质量直接影响到算法的收敛速度，适应度值高的种群能够很快地收敛到接近最优解的解，因此他提出了一种新的初始化种群的方法，以增高初始种群的适应度值；Fernando Jose Mateus da Silva(2009)提出了将局部最优搜索融入遗传算法中的新算法，提高了算法的准确度；胡桂武(2004)提出了一种基于遗传算法与星比对算法的多序列比对混合算法；司秀华(2006)提出了一种多搜索策略的多生物序列比对自适应遗传算法，这种算法是通过调整遗传算法中的交叉率和变异率从而避免算法找到局部最优而提出的一种算法。以上种种改进使遗传算法有了很多不同的算法模型和算法流程，但其大概流程都没有脱离原有的算法流程。在本书第 4 章中的遗传算法改进就是基于基本结构的优化改进。

2. 混合遗传算法

近年来，许多学者提出通过将遗传算法与其他优化算法相结合的方法，来避免基本遗传算法所出现的一些缺点。梁旭等将遗传算法和模拟退火算法结合，提出遗传退火混合算法；梁艳春等在隐马尔科夫(HMM)模型中结合了粒子群优化和模拟退火进行学习，并且在训练过程中结合人工免疫策略；张维存等将蚁群算法和遗传算法结合，提出主从递减结构的蚁群遗传算法；廖波等也将蚁群算法和遗传算法结合起来解决多序列比对问题；张维梅(2008)提出了一种基于遗传算法和蚁群算法的多重序列比对算法，这种算法是将蚁群算法作为局部搜索的一种算法。还有其他比较常见的混合遗传算法，如免疫遗传算法、小生境遗传算法、量子遗传算法、DNA 遗传算法、病毒遗传算法、并行混合遗传算法等。

3.2　粒子群优化算法

3.2.1　基本粒子群优化算法

粒子群优化(PSO)算法最早是由美国心理学家 James Kennedy 和电器工程师 Russell Eberhart 于 1995 年提出的一种基于群体智能的优化算法[3]，该算法源于对人工生命和鸟群、鱼群等生物种群觅食行为的研究。设想这样一个场景：一群鸟在随机搜寻食物，在这个区域里只有一块食物，所有的鸟都不知道食物在哪里，但是它们知道当前的位置离食物还有多远，那么找到食物的最优策略是什么？最简单有效的方法就是搜寻目前离食物最近的鸟的周围区域。

PSO 算法就从这种生物种群行为特性中得到启发并有效地用于求解复杂优化问题。在 PSO 系统中，每个优化问题的潜在解都可以想象成 N 维搜索空间上的一个点，称之为“粒子”(particle)，而所有的粒子都有一个被目标函数决定的适应值(fitness value)，即目标

函数值。每个粒子在搜索空间中以一定的速度飞行，这个速度根据它本身的飞行经验和其他粒子的飞行经验来动态调整。通常粒子将追随当前的最好粒子，并经逐代搜索，最终得到最优解。在每一代中，粒子将跟踪两个最好位置，一是粒子本身迄今找到的最好位置，称为个体最好(personal best，pbest)位置；另一个为整个粒子群迄今为止找到的最好位置，称为全局最好(global best，gbest)位置。

其搜索过程数学描述为：假设在一个 N 维的目标搜索空间中，由 M 个代表潜在问题解的粒子组成群体 $X=\{X_1,X_2,\cdots,X_M\}$，在 t 时刻，第 i 个粒子位置为 $X_i(t)=[X_{i,1}(t),X_{i,2}(t),\cdots,X_{i,N}(t)]$,速度为 $V_i(t)=[V_{i,1}(t),V_{i,2}(t),\cdots,V_{i,N}(t)]$，$i=1,2,\cdots,M$。个体最好位置表示为 $P_i(t)=[P_{i,1}(t),P_{i,2}(t),\cdots,P_{i,N}(t)]$，群体的全局最好位置为 $G(t)=[G_1(t),G_2(t),\cdots,G_N(t)]$，且 $G(t)=P_g(t)$，其中 g 为处于全局最好位置粒子的下标，$g\in\{1,2,\cdots,M\}$。

对于最小化问题，目标函数值越小，对应的适应值越好。粒子 i 的个体最好位置 pbest 由式(3.1)确定：

$$P_i(t)=\begin{cases}X_i(t) & f[X_i(t)]<f[P_i(t-1)]\\ P_i(t-1) & f[X_i(t)]\geqslant f[P_i(t-1)]\end{cases}\tag{3.1}$$

群体的全局最好位置 gbest 由式(3.2)和式(3.3)确定：

$$g=\arg\min_{1\leqslant i\leqslant M}\{f[P_i(t)]\tag{3.2}$$

$$G(t)=P_g(t)\tag{3.3}$$

有了以上定义，基本粒子群算法的进化方程可描述为

$$\begin{aligned}V_{i,j}(t+1)=V_{i,j}(t)&+c_1\cdot r_{1,j}(t)\cdot[P_{i,j}(t)-X_{i,j}(t)]\\&+c_2\cdot r_{2,j}(t)\cdot[G_j(t)-X_{i,j}(t)]\end{aligned}\tag{3.4}$$

$$X_{i,j}(t+1)=X_{i,j}(t)+V_{i,j}(t+1)\tag{3.5}$$

式中，$j(j=1,2,\cdots,N)$为粒子的第 i 维；N 为搜索空间维数；下标 $i(=1,2,\cdots,M)$表示第 i 个粒子；M 为群体规模；t 为迭代代数；c_1 和 c_2 为加速因子；r_1 和 r_2 为[0,1]区间上均匀分布的随机数。

从上述粒子进化方程可以看出，c_1 调节粒子飞向自身最好位置

方向的步长，c_2 调节粒子向全局最好位置方向的步长。为了减少在进化过程中，粒子离开搜索空间的可能性，$V_{i,j}$ 通常限于一定的范围内，即 $V_{i,j} \in [-V_{\max}, V_{\max}]$。如果问题的搜索空间限定在 $X_{i,j} \in [-X_{\max}, X_{\max}]$ 内，则可设定 $V_{\max} = k \cdot X_{\max}$，$0.1 \leqslant k \leqslant 1.0$。

粒子位置更新公式(3.5)以速度为步长进行更新；粒子速度更新公式(3.4)可看成由三部分组成：①粒子先前迭代步的速度；②个体认知部分，表示粒子自身的思考，使粒子具有足够强的全局搜索能力，避免局部极小；③社会认知部分，体现了粒子间的信息共享。在这三部分的共同作用下，粒子根据历史经验并利用信息共享机制，不断调整自己的位置，以期望找到问题的最优解。

基本粒子群算法的初始化过程为：

(1) 设定群体规模 M。

(2) 对任意 i，j，在 $[-X_{\max}, X_{\max}]$ 内服从均匀分布产生 $X_{i,j}$。

(3) 对任意 i，j，在 $[-V_{\max}, V_{\max}]$ 内服从均匀分布产生 $V_{i,j}$。

(4) 对任意 i，设 $P_i = X_i$。

基本粒子群算法的描述如下：

(1) 依照初始化过程，对粒子群的随机位置和速度进行初始化。

(2) 计算每个粒子的适应值。

(3) 对于每个粒子，将其适应值与所经历的最好位置 P_i 的适应值进行比较，若优于 P_i 的适应值，则将其作为当前的最好位置。

(4) 对于每个粒子，将其适应值与粒子群所经历全局最好位置 G 的适应值进行比较，若优于 G 的适应值，则将其作为当前的全局最好位置。

(5) 根据式(3.4)、式(3.5)对粒子的速度和位置进行更新。

(6) 如未达到结束条件[通常为足够好的适应值或达到一个预设最大代数($t_{\max}$)]，则返回步骤(2)。

3.2.2　带惯性权重 w 的粒子群优化算法

对于不同的问题，如何确定局部搜索能力与全局搜索能力的比

例关系，对于其求解过程非常重要。甚至对于同一问题而言，进化过程中也要求不同的比例。为此，Yuhui Shi 提出了带有惯性权重的改进粒子群算法。其进化方程为

$$V_{i,j}(t+1)=w\bullet V_{i,j}(t)+c_1\bullet r_{1,j}(t)\bullet[P_{i,j}(t)-X_{i,j}(t)] \\ +c_2\bullet r_{2,j}(t)\bullet[G_j(t)-X_{i,j}(t)] \tag{3.6}$$

$$X_{i,j}(t+1)=X_{i,j}(t)+V_{i,j}(t+1) \tag{3.7}$$

当惯性权重为 w=1 时，式(3.6)与基本粒子群算法的速度进化方程相同，从而表明带惯性权重的粒子群算法是基本粒子群算法的扩展。文献建议 w 的取值范围为[0, 1.4]，但实际结果表明当 w 取[0.8,1.2]时，算法的收敛速度更快，而当 w>1.2 时，算法则由于收敛速度慢，较多地陷入次优解。

惯性权重 w 表明粒子原先的速度能在多大程度上得到保留。假设粒子的初始速度非零，当 $c_1=c_2=0$ 且 $w>0$ 时，粒子将会加速直至 V_{max}；当 $w<0$，则粒子将会减速直至 0；当 $c_1,c_2\neq 0$ 时，情况比较复杂，但实验结果表明，w=1 时效果要好一些。

惯性权重类似模拟退火中的温度，较大的 w 有较好的全局搜索能力，而较小的 w 则有较强的局部搜索能力。因此，随着迭代次数的增加，惯性权重 w 应不断减少，从而使得粒子群算法在初期有较强的全局收敛能力，而在晚期具有较强的局部收敛能力。惯性 w 权重满足

$$w(t)=0.9-\frac{t}{t_{max}}\times 0.4 \tag{3.8}$$

式中，t_{max} 为最大迭代步数(或称为最大进化代数)。

这样，将惯性权重 w 看作迭代次数的函数，可从 0.9 到 0.4 线性减少，从对四个主要测试函数的测试结果来看，效果很好。

目前，有关 PSO 算法的研究大多以带惯性权重的 PSO 算法为基础进行扩展和修正。为此，在大多数文献中将带惯性权重的 PSO 算法称为 PSO 算法的标准版本或简称标准 PSO(SPSO)；而将基本 PSO 算法称为 PSO 的初始版本。

3.2.3　带收缩因子 χ 的粒子群优化算法

Clerc 在他的研究中提出了收缩因子的概念。该方法描述了一种选择 w、c_1 和 c_2 的值的方法，以确保算法收敛。通过正确地选择这些控制参数，就没有必要将 $V_{i,j}$ 的值限制在$[-V_{max},V_{max}]$中。接下来首先讨论一个与带有收缩因子的粒子群算法相关的收敛模式特例。

一个与某个收敛模式相符合的改进了的速率方程式以下述形式提出：

$$\begin{aligned}V_{i,j}(t+1)=\chi\cdot\{V_{i,j}(t)+c_1\cdot r_{1,j}(t)\cdot[P_{i,j}(t)-X_{i,j}(t)]+\\ c_2\cdot r_{2,j}(t)\cdot[G_j(t)-X_{i,j}(t)]\}\end{aligned} \tag{3.9}$$

式中

$$\chi=\frac{2}{\left|2-l-\sqrt{l^2-4l}\right|}\quad(l=c_1+c_2,l>4) \tag{3.10}$$

设 $c_1=c_2=2.05$，将 $l=c_1+c_2=4.1$ 代入式(3.10)，得出 $\chi=0.7298$ 并代入式(3.9)，结果为

$$\begin{aligned}V_{i,j}(t+1)=0.7298\{V_{i,j}(t)+2.05\times r_{1,j}(t)\cdot[P_{i,j}(t)-X_{i,j}(t)]+\\ 2.05\times r_{2,j}(t)\cdot[G_j(t)-X_{i,j}(t)]\}\end{aligned} \tag{3.11}$$

因为 2.05×0.7298=1.4961，所以这个方程式与在改进的 PSO 速率更新方程使用 $c_1=c_2=1.4961$ 和 $w=0.7298$ 所得到的方程式是等价的。

Eberhart 和 Shi 将分别利用 V_{max} 和收缩因子来控制粒子速度的两种算法性能做了比较。结果表明，后者比前者通常具有更好的收敛速率。然而在有些测试函数的求解过程中，使用收缩因子的 PSO 在给定迭代次数内无法达到全局极值点。按照 Eberhart 和 Shi 的观点，这是由于微粒偏离所期望的搜索空间太远而造成的。为了降低这种影响，他们建议在使用收缩因子时首先对算法进行限定，如设参数 $V_{max}=X_{max}$，或者预先设置搜索空间的大小。这样可以改进算法对所有测试函数的求解性能，不管是在收敛速率方面还是在搜索能力方面。

至于其他的改进算法，第 2 章已有详细讨论，这里不再赘述。

3.3　量子粒子群优化算法

3.3.1　δ 势阱模型的建立

在量子空间中，粒子的速度和位置是不能同时确定的，因此粒子的状态必须用所谓的波函数 $\psi(X,t)$ 来描述，其中 $X=(x,y,z)$ 是粒子在三维空间中的位置向量。波函数的物理意义是：波函数模的平方是粒子在空间某一点出现的概率密度，即

$$|\psi|^2 \mathrm{d}x\mathrm{d}y\mathrm{d}z = Q\mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{3.12}$$

式中，Q 为概率密度函数。

当然，这个概率分布密度函数满足以下归一化条件：

$$\int_{-\infty}^{+\infty} |\psi|^2 \mathrm{d}x\mathrm{d}y\mathrm{d}z = \int_{-\infty}^{+\infty} Q\mathrm{d}x\mathrm{d}y\mathrm{d}z = 1 \tag{3.13}$$

在量子空间中粒子运动的动力学方程是 Schrödinger 方程，即

$$i\hbar \frac{\partial}{\partial t}\psi(X,t) = \hat{H}\psi(X,t) \tag{3.14}$$

式中，$\hat{H}$ 是哈密顿算子；$\hbar$ 称为普朗克常数。

哈密顿算子 $\hat{H}$ 具有以下形式：

$$\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + V(X) \tag{3.15}$$

式中，m 是粒子的质量；$V(X)$ 是粒子所在的势场。

现在假定粒子群系统是一个量子粒子系统，每一个粒子具有量子行为，由波函数来描述其状态。根据 PSO 算法中粒子收敛行为的分析，必然存在以点 p_i 为中心某种形式的吸引势。为简单起见，先考虑单个粒子在一维空间中运动的情形，并且将 p_i 记为 p，粒子的位置为 X。在 p 点建立一维 δ 势阱，其势能函数表示为

$$V(x) = -\gamma\delta(X-p) = -\gamma\delta(Y) \tag{3.16}$$

式中，$Y = X - p$；m 为粒子的质量。

因此该问题的哈密顿算子为

$$\hat{H} = -\frac{\hbar^2}{2m} \cdot \frac{\mathrm{d}^2}{\mathrm{d}Y^2} - \gamma\delta(Y) \tag{3.17}$$

粒子在 δ 势阱中的定态 Schrödinger 方程为

$$\frac{\mathrm{d}^2\psi}{\mathrm{d}Y^2} + \frac{2m}{\hbar^2}[E + \gamma\delta(Y)]\psi = 0 \tag{3.18}$$

式中，E 是粒子的能量。

于是得到以下定理。

定理 3.1：粒子在以 p 点为中心的一维 δ 势阱中运动，对应的定态 Schrödinger 方程的解为

$$\psi(Y) = \frac{1}{\sqrt{L}} \mathrm{e}^{-|Y|/L} \tag{3.19}$$

式中，$L = 1/\beta = \hbar^2/m\gamma$。

证明：对式(3.18)两边求积分 $\int_{-\varepsilon}^{+\varepsilon} \mathrm{d}Y$，当 $\varepsilon \to 0^+$ 时可得

$$\psi'(0^+) - \psi'(0^-) = -\frac{2m\gamma}{\hbar^2}\psi(0) \tag{3.20}$$

当 $Y \neq 0$，式(3.18)可写为

$$\frac{\mathrm{d}^2\psi}{\mathrm{d}Y^2} - \beta^2\psi = 0 \tag{3.21}$$

式中，

$$\beta = \sqrt{-2mE/\hbar} \quad (E < 0)$$

为了满足束缚态条件：

$$|Y| \to \infty, \quad \psi \to 0 \tag{3.22}$$

式(3.21)的解必须具有以下形式：

$$\psi(Y) \approx \mathrm{e}^{-\beta|Y|} \qquad (Y \neq 0) \tag{3.23}$$

由于波函数必须满足束缚态条件(3.22)，因此式(3.21)的解为

$$\psi(Y)=\begin{cases}Ae^{-\beta Y} & Y>0\\ Ae^{\beta Y} & Y<0\end{cases} \tag{3.24}$$

式中，A 是归一化常数。

根据条件式(3.20)可得

$$-2A\beta=-\frac{2m\gamma}{\hbar^2}A \tag{3.25}$$

这样得到

$$\beta=m\gamma/\hbar^2 \tag{3.26}$$

以及

$$E=E_0=-\frac{\hbar^2\beta^2}{2m}=-\frac{m\gamma^2}{2\hbar^2} \tag{3.27}$$

函数 $\psi(Y)$ 须满足归一化条件以决定常数 A，即

$$\int_{-\infty}^{+\infty}|\psi(Y)|^2\mathrm{d}Y=|A|^2/\beta=1 \tag{3.28}$$

于是得到 $|A|=\sqrt{\beta}$，$L=1/\beta=\dfrac{\hbar^2}{m\gamma}$ 是 δ 势阱的特征长度。代入式(3.24)，则归一波函数表示为

$$\psi(Y)=\frac{1}{\sqrt{L}}e^{-|Y|/L} \tag{3.29}$$

相应的概率密度函数 Q 为

$$Q(Y)=|\psi(Y)|^2=\frac{1}{L}e^{-2|Y|/L} \tag{3.30}$$

概率分布函数 F 为

$$F(Y)=1-e^{-2|Y|/L} \tag{3.31}$$

证毕。

3.3.2 粒子的基本进化方程

从定理 3.1 的证明过程，得到了粒子在一维 δ 势阱的量子束缚态波函数和相应的位置概率分布函数。在实际的算法设计中，为评价适应值(目标函数值)，需要了解粒子精确的位置。而量子状态函数 $\psi(Y)$ 仅仅给出粒子出现在相对于 p 点位置 Y 的概率密度函数 $|\psi(Y)|^2$ 或 $Q(Y)$。因此必须给出粒子的位置，将量子状态塌缩到经典状态。此时，可以通过蒙特卡罗随机模拟的方式来测量粒子的位置，这种方法好比对粒子进行拍照，在按快门的一瞬间粒子的位置就被定格在照片上了。这里采用的蒙特卡罗方法又称为逆变换法，这是针对概率分布函数形式比较简单的情况下最常用的随机模拟方法。推导过程如定理 3.2。

定理 3.2：粒子在以 p 点为中心的一维 δ 势阱中运动，其位置由以下随机方程确定，即

$$X = p \pm \frac{L}{2}\ln(1/u) \tag{3.32}$$

式中，$L = 1/\beta = \hbar^2/m\gamma$；$u$ 为区间(0,1)上的均匀分布随机数，即 $u \sim U(0,1)$。

证明：令 v 是在区间(0,1)上均匀分布的随机数，即

$$v \sim U(0,1) \tag{3.33}$$

用 v 代替式(3.31)中的左边，即令

$$1 - v = 1 - F(Y) \tag{3.34}$$

由于 $1 - v \sim U(0,1)$，令 $u = 1 - v$，则有 $u \sim U(0,1)$。于是可以得到

$$u = \mathrm{e}^{-2|Y|/L} \tag{3.35}$$

用逆变换求出 Y

$$Y = \pm\frac{L}{2}\ln(1/u) \tag{3.36}$$

由于 $Y = X - p$，因此可以测量粒子的位置的随机方程：

$$X = p \pm \frac{L}{2}\ln(1/u) \tag{3.37}$$

式中，u 是(0,1)区间的均匀分布的随机数。

证毕。

式(3.32)是量子行为粒子群算法的基本进化方程，其中 L 是 δ 势阱的特征长度，是进化方程中最重要的参量。

3.3.3　QPSO 算法的流程

在一个 N 维的目标搜索空间中，QPSO 算法由 M 个代表潜在问题解的粒子组成群体 $X=\{X_1,X_2,\cdots,X_M\}$，在 t 时刻，第 i 个粒子位置为 $X_i(t)=[X_{i,1}(t), X_{i,2}(t),\cdots,X_{i,N}(t)]$，$i=1,2,\cdots,M$，粒子没有速度向量。个体最好位置表示为 $P_i(t)=[P_{i,1}(t),P_{i,2}(t),\cdots,P_{i,N}(t)]$，群体的全局最好位置为 $G(t)=[G_1(t),G_2(t),\cdots,G_N(t)]$，且 $G(t)=P_g(t)$，其中 g 为处于全局最好位置粒子的下标，$g\in\{1,2,\cdots,M\}$。

对于最小化问题，目标函数值越小，对应的适应值越好。粒子 i 的个体最好位置 pbest 由下式确定：

$$P_i(t)=\begin{cases} X_i(t) & f[X_i(t)]<f[P_i(t-1)] \\ P_i(t-1) & f[X_i(t)]\geqslant f[P_i(t-1)] \end{cases} \tag{3.38}$$

群体的全局最好位置 gbest 由式(3.2)和式(3.3)确定：

$$g=\arg\min_{1\leqslant i\leqslant M}\{f[P_i(t)]\} \tag{3.39}$$

$$G(t)=P_g(t) \tag{3.40}$$

令

$$p_{i,j}(t)=\phi_j(t)\cdot P_{i,j}(t)+[1-\phi_j(t)]\cdot G_j(t)\quad \phi_j(t)\sim U(0,1) \tag{3.41}$$

则粒子的更新方程为

$$X_{i,j}(t+1)=p_{i,j}(t)\pm\alpha\cdot\left|p_{i,j}(t)-X_{i,j}(t)\right|\cdot\ln\left[1/u_{i,j}(t)\right]\quad u_{i,j}(t)\sim U(0,1) \tag{3.42}$$

或

$$X_{i,j}(t+1)=p_{i,j}(t)\pm\alpha\cdot\left|C_j(t)-X_{i,j}(t)\right|\cdot\ln\left[1/u_{i,j}(t)\right]\quad u_{i,j}(t)\sim U(0,1) \tag{3.43}$$

式中，

$$\begin{aligned} C(t) &= (C_1(t), C_2(t), \cdots, C_n(t)) = \frac{1}{M}\sum_{i=1}^{M} P_i(t) \\ &= \left(\frac{1}{M}\sum_{i=1}^{M} P_{i,1}(t), \frac{1}{M}\sum_{i=1}^{M} P_{i,2}(t), \frac{1}{M}\sum_{i=1}^{M} P_{i,n}(t)\right) \end{aligned} \tag{3.44}$$

以下是 QPSO 算法的执行过程：

(1) 在问题空间中初始化粒子群中粒子的位置。

(2) 根据式(3.44)计算粒子群的平均最优位置。

(3) 计算粒子的当前适应值，并与前一次迭代的适应值比较，如果当前适应值小于前一次迭代的适应值，则根据粒子的位置更新为粒子当前的位置，即如果 $f[X_i(t+1)] < f[P_i(t)]$，则 $P_i(t+1) = X_i(t+1)$。

(4) 计算群体当前的全局最优位置，即 $G(t) = P_g(t)$， $g = \arg\min_{1 \leqslant i \leqslant M} \{f[P_i(t)]$。

(5) 比较当前全局最优位置与前一次迭代的全局最优位置，如果当前全局最优位置的位置较好，则群体的全局最优位置更新为它的值。

(6) 对粒子的每一维，根据式(3.41)计算得到一个随机点的位置。

(7) 根据式(3.42)或式(3.43)计算粒子的新的位置。

(8) 重复步骤(2)～(7)，直至满足一定的循环结束条件。

3.3.4　QPSO 算法的收敛性分析

首先给出一些判别算法收敛的定义：

定义 3.1： 局部搜索算法——只能保证搜索到目标函数的局部最优解的算法。

定义 3.2： 全局搜索算法——能够保证搜索到目标函数的全局最优解的算法。

Solis 和 Wets 研究了随机搜索算法的收敛性，特别是研究了抽

象随机搜索算法，并给出了算法属于全局搜索算法还是仅属于局部搜索算法的标准。

1. 全局搜索算法的收敛准则

定义 3.3：对于最优化问题给定一个目标函数 f，它的解空间是从 $\mathbf{R}^n$ 到 $\mathbf{R}$ 的映射，S 是 $\mathbf{R}^n$ 的一个子集。在 S 中寻找一个点 z，能够使得函数 f 的值最小化或者至少能够生成一个函数 f 在 S 上的可接受的下确界。

这个定义给出了一个全局优化算法在给定目标函数和搜索空间的情况下必须能够产生输出。能够完成这个任务的最简单随机算法是基本随机搜索算法。在第 t 步迭代，算法需要一个三元组的概率空间($\mathbf{R}^n$, B, μ_t)，其中 μ_t 是 B 上的概率测度(对应于 $\mathbf{R}^n$ 上的分布函数)，B 是 $\mathbf{R}^n$ 的子集组成的 Borel σ 域。定义概率测度 μ_t 的支撑集为 M_t，则 M_t 就是在概率测度 μ_t 下以测度 1 最小闭子集。

随机算法的基本框架如下：

(1) 随机选择初始点 $z_0 \in S$，并设置 t=0。

(2) 在样本空间($\mathbf{R}^n$,B,μ_t)上生成 ξ_t。

(3) 计算 $z_{t+1} = D(z_t, \xi_t)$，选择 μ_{t+1}，令 t=t+1，并转(1)。

其中 D 是可以在问题空间产生一个解的函数(或算子)，能保证 D 所产生的新个体优于当前个体，因此，随机算法应满足以下假设：

假设 3.1：$f(D(z,\xi)) \leqslant f(z)$ 且 $\xi \in S$，则

$$f(D(z,\xi)) \leqslant f(\xi) \tag{3.45}$$

不同的 D 就代表了不同的具体算法，但是都必须满足假设 3.1 以保证优化算法的正确运行。

任何算法的全局收敛意味着序列 $\{f(z_t)\}_{t=1}^{\infty}$ 应收敛于函数 f 在 S 上的下确界。一个病态的例子是一个函数的最小值点在它的间断点的话，其 Lebesgue 测度为 0，对于任何随机算法而言，几乎不可能搜索到全局最优解，如下面的函数：

$$f=\begin{cases}x^2 & \forall x\neq 1\\ -10 & x=1\end{cases} \tag{3.46}$$

函数 f 的最优值为 $f(1)$=-10。

因此为了避免出现上面所述的病态情况，将搜索的目标变为搜索本质下确界 φ：

$$\varphi=\inf\left(k: v[z\in S \mid f(z)<k]>0\right) \tag{3.47}$$

式中，$v[A]$ 是在集合 A 上的 Lebesgue 测度。

式(3.47)表明搜索空间的一个子集必定有不止一个点使函数值以任意的方式接近 φ，这样 φ 就是从非零的 v 测度集上得到的函数值的下界。典型情况是，$v[A]$ 是集合 A 的一个 n 维空间。通过定义一个新的下界使得在搜索空间内始终有一个任意小的非空集合围绕着包含 S 的空间，可以使得 φ 避免了上面提到的病态的情形。这种方法使得算法可以接近下界而无须遍历 S 中的每一个点。

定义算法的 ε 可接受区域为

$$R_\varepsilon=\left\{z\in S \mid f(z)<\varphi+\varepsilon\right\} \tag{3.48}$$

式中，$\varepsilon>0$。

如果算法发现 R_ε 中的点，则称算法找到了误差为 ε 的可接受点。

一个局部收敛算法是指对于测度序列 μ_t，支撑集序列 M_t，除了有限个集合外，都有界且 $M_t\subset S$。因此，局部收敛算法的支撑集满足 $v[S\cap M_t]<v[S]$，也就说明搜索空间的部分区域可能一直都不会被访问到，即一个真正的全局收敛算法应该满足下面的假设：

假设 3.2：对于 S 的任意 Borel 子集 A，若其测度 $v[A]$>0，则有

$$\prod_{t=0}^{\infty}\left(1-\mu_t[A]\right)=0 \tag{3.49}$$

式中，$\mu_t[A]$ 是由测度 μ_t 所得到的 A 的概率。

这就意味着对于位置测度为 v 的任意一个 A 的子集来说，如果采用随机取样的方法(如上面提到的 ζ_t)，那么它重复错过集合 A 的

概率必定为 0。由于$R_\varepsilon \subset S$，所有在可接受区域取得点的概率肯定是非零值。

利用假设 3.1 和 3.2 可以给出随机算法为全局收敛算法的充要条件。

定理 3.3：假设目标函数f为可测函数，区域S是$\mathbf{R}^n$的可测子集，假设 3.1 和 3.2 满足，设$\{z_t\}_{t=0}^{\infty}$为算法生成的解序列，可得

$$\lim_{t\to+\infty} P[z_t \in R_\varepsilon] = 1 \tag{3.50}$$

式中，$P[z_t \in R_\varepsilon]$是第t步算法生成的解$z_t \in R_\varepsilon$的概率。

通过全局搜索的定理可以找到满足假设 3.1 和假设 3.2 的全局优化算法。

2. 局部搜索算法的收敛准则

上面介绍了一个算法满足随机全局搜索算法需要的条件。虽然基本随机搜索算法可以满足相应的条件，但是由于搜索速度太慢而很难作为一个实际的算法。局部搜索算法具有较快的收敛速度，但以牺牲找不到全局解为代价。本部分将会给出局部搜索算法的收敛准则。

对于不能满足假设 3.2 的算法来说，可以定义局部搜索条件如下：

假设 3.3：对于任意$z_0 \in S$，存在$\gamma > 0$，$0 < \eta \leqslant 1$，使得：

$$\mu_t[(dist(D(z,\zeta),R_\varepsilon) < dist(z,R_\varepsilon) - \gamma) \text{ or } (D(z,\zeta) \in R_\varepsilon)] \geqslant \eta \tag{3.51}$$

对于所有的t和集合$L_0 = \{z \in S \mid f(z) \leqslant f(z_0)\}$的$z$均成立。其中，$dist(z,A)$表示点$z$与集合$A$之间的距离，定义为

$$dist(z,A) = \inf_{b\in A} dist(z,b) \tag{3.52}$$

因此一个局部收敛算法可以定义为，存在一个非零数η，算法在每一个迭代步后可以将点z在移动最短的距离γ后靠近最优区域，或者点z已经在最优区域的情况下以概率大于或等于η。结合假设

3.1、3.2，给出随机算法为局部收敛算法的一个充要条件。

定理 3.4：假定目标函数 f 为可测函数，区域 S 为 $\mathbf{R}^n$ 下的可测子集，并且假设 3.1 和 3.3 满足，设$\{z_t\}_{t=0}^{\infty}$为算法所生成的解序列，则 $\lim\limits_{t\to+\infty} P[z_t \in R'_\varepsilon]=1$，其中，$P[z_t \in R'_\varepsilon]$是第 t 步算法生成的解 $z_t \in R'_\varepsilon$ 的概率，区域 R'_ε 表示某一局部极值点的 ε 可接受区域。

3. QPSO 算法的全局收敛性

Van den Bergh 指出 PSO 算法既不是一个局部收敛算法也不是一个全局收敛算法。为了证明 QPSO 的全局收敛性，将 QPSO 算法置于全局随机搜索算法的框架中，以定理 3.3 的结论进行证明，因此需要证明 QPSO 算法能满足假设 3.1 和假设 3.2。

引理 3.1：QPSO 算法满足假设 3.1。

证明：根据式(3.1)，函数 D(在假设 3.1 中提到)在 QPSO 算法中的描述可以定义为

$$D(P_t, X_{it}) = \begin{cases} P_{g,k} & f(app(x_{i,t})) \geqslant f(P_{g,k}) \\ app(x_{i,t}) & f(app(x_{i,t})) < f(P_{g,k}) \end{cases} \tag{3.53}$$

式中，$app(x_{i,t})$表示 app 的具体应用，它通过 QPSO 算法的迭代更新执行，由式(3.54)与式(3.55)定义。将 $x_i(t)$ 用 $x_{i,t}$ 来表示，t 即为算法的迭代次数。上面的内容已经写明对每一次迭代 n 的依赖关系。序列 $\{P_{g,l}\}_{l=0}^{t}$ 是所有粒子从开始到第 t 次迭代步(包括第 t 次迭代)所到达的最好的位置序列。

可以将 $x_{i,t+1}$ 作为连续应用函数 g 得到的计算结果，即

$$x_{i,t+1} = app(x_{i,t}) \tag{3.54}$$

app 是一个矢量函数，用 $app(X_{i,t})_j$ 表示函数 app 的第 j 维，具体表达方式为

$$app(X_{i,t})_j = \phi_{i,j,t} P_{i,j,t} + (1-\phi_{i,j,t}) P_{g,j,t} + \alpha \left| C_{j,t} - x_{i,j,t} \right| \ln\left(\frac{1}{u_{i,j,t}}\right) \tag{3.55}$$

式中，$C_{j,t}$ 是平均最优位置 C 在第 t 次迭代的第 j 维；$\phi_{i,j,t}$ 是服从(0,1)均匀分布的随机序列。

按照算法的定义，序列 $P_{g,t}$ 是单调，因此 D 的定义显然是符合假设 3.1 的。证毕。

引理 3.2：QPSO 算法满足假设 3.2。

证明：在上面的阐述中，在任意一个迭代步 t，第 i 个粒子的第 j 维的概率密度函数为

$$Q(x_{i,j,t}) = \frac{1}{L_{i,j,t}} \exp\left(-2\left|x_{i,j,t} - p_{i,j,t}\right| / L_{i,j,t}\right) \tag{3.56}$$

粒子 i 的概率密度函数可以表示为

$$Q(x_{i,t}) = \prod_{j=1}^{n} \frac{1}{L_{i,j,t}} \exp\left(-2\left|x_{i,j,t} - p_{i,j,t}\right| / L_{i,j,t}\right) \tag{3.57}$$

定义 $\mu_{i,t}$ 为对应于 n 维双指数概率分布。因此，对于 S 的任意 Borel 子集 A，当满足 $v[A] > 0$ 时，可以得到

$$\mu_{i,t}[A] = \int_A [\prod_{j=1}^{n} \frac{1}{L_{i,j,t}} \exp(-2\,|\,x_{i,j,t} - p_{i,j,t}\,| / L_{i,j,t})] \mathrm{d}x_{i,1,t} \mathrm{d}x_{i,2,t} \cdots \mathrm{d}x_{i,n,t} \tag{3.58}$$

$$M_{i,t} = \mathbf{R}^n \supset S \tag{3.59}$$

式中，$M_{i,t}$ 是 $\mu_{i,t}$ 在样本空间的支撑，并且 $A \supset M_{i,t}$。

因此，可以得到

$$0 < \mu_{i,t}[A] < 1 \tag{3.60}$$

所有粒子支撑的并集为

$$M_t = \cup_{i=1}^{s} M_{i,t} = \mathbf{R}^n \supset S \tag{3.61}$$

式中，M_t 是分布 μ_t 的支撑。

由 μ_t 产生的对 A 的概率测度可以由下式计算得到

$$\mu_t[A] = 1 - \prod_{i=1}^{s} (1 - \mu_{i,t}[A]) \tag{3.62}$$

通过式(3.60)，可得

$$0 < \mu_t[A] < 1 \text{ for } \quad t = 1,2,\cdots \tag{3.63}$$

因此可得

$$\prod_{t=0}^{\infty}(1-\mu_t[A]) = 0 \tag{3.64}$$

这就表明 QPSO 算法满足假设 3.2。证毕。

定理 3.5：QPSO 算法是一个全局收敛算法。

证明：由于 QPSO 算法满足假设 3.1 和假设 3.2，由定理 3.3，可以得到 QPSO 算法是一个全局收敛算法的结论。证毕。

3.4　QPSO 算法的改进——基于选择操作的 QPSO 算法

3.4.1　引言

QPSO 算法是一个较有发展前景的全局优化算法，在许多实际应用领域中优于 PSO 算法。首先，由于在算法中引入了双指数分布使得算法成为一个全局优化算法。此外，在算法中引入了平均最好位置 C 也对 QPSO 算法的性能改进较大。在 PSO 算法中，每一个粒子都相互独立地收敛到全局最好位置 P_g，但是，在 QPSO 算法中，由于引入了平均最好位置 C 作为参考点，每个粒子不能不顾其他粒子而独立地向 P_g 汇聚，粒子间存在相互等待的过程，如图 3.4 所示，右上角的两个粒子远离 P_g，称为落后粒子(lagged particles)。原因如下：粒子当前的位置和粒子群的平均位置 C 之间的距离决定了下一代粒子位置的分布。假如有落后粒子，平均最好位置 C 会被落后粒子的个体最好位置拉离 P_g，当落后粒子趋向 P_g 时，C 会以较慢的速度收敛于 P_g，P_g 附近粒子的位置与 C 之间的距离不会快速地减小，因此它们的收敛速度减慢了，并且让它们临时地在 P_g 附近进行全局

搜索直到落后粒子靠近 P_g 。因此，引入了平均最好位置 C 的 QPSO 算法不会放弃任何一个落后粒子，这时的群体更具智能化并且协同工作能力更强。

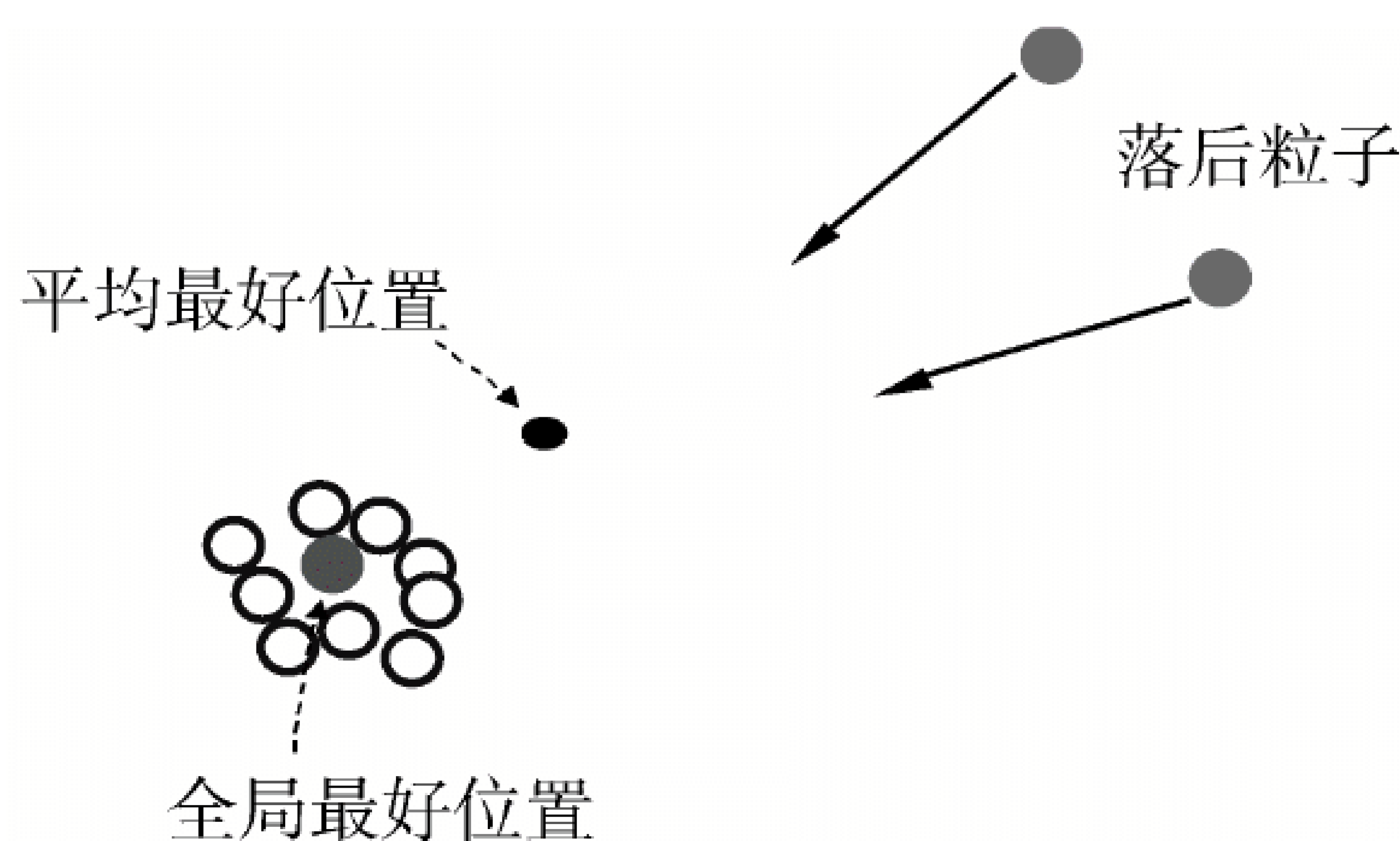


图 3.4　QPSO 算法中粒子的等待效应

与 PSO 算法一样，QPSO 算法中的所有粒子最后会收敛到全局最好位置 P_g ，这就使 QPSO 算法与遗传算法相比有较快的收敛速度，并且得到的最优解精度比较高。然而，粒子的记忆性和它们对 P_g 的收敛性很容易导致早熟。如图 3.4 所示，当大多数粒子聚集到全局最好位置 P_g 附近时，落后的粒子也会被拉向 P_g 。如果这时 P_g 在全局最优解的邻域内，则粒子向 P_g 的收敛可以增强对该邻域的局部搜索，从而提高解的精度。但如果 P_g 位于局部最优解或次优解的邻域内，并且离全局最优解较远时，粒子向 P_g 的收敛可能导致早熟。如果全局最优解或更好的解位于落后粒子目前所在区域，由于落后粒子以较大的概率出现在 P_g 附近，很容易错过全局最优解或更好的解。随着落后粒子趋向 P_g ，所有粒子都开始做更小范围内的局部搜索，找到全局最优解或更好解的概率就越来越小，这样就产生了早熟收敛。

由于在 QPSO 算法中会遇到上面的早熟收敛的现象，所以提出了在 QPSO 算法中加入两种选择策略来避免早熟收敛的现象。这种选择操作与 Angeline 提出的用于 PSO 算法的选择操作不同，它是施加在全局最好位置 P_g 上的，这时吸引点 p_i 不是由 P_i 和 P_g 来决定，而

是由 P_i 和某个随即选择出的粒子的个体最优位置决定。这两种选择策略来源于遗传算法中的锦标赛选择(tournament selection)和轮盘赌选择(roulette wheel selection)。采用上述选择策略有下面的两个优势，一是如果所有的粒子聚集在 P_g 附近，那么这些粒子会被随机选择的粒子推开，使得粒子群早熟收敛的概率就会越来越小；二是如果随机选择出的粒子的个体最优位置比较接近于全局最优位置，那么向它靠近的粒子将会有更大的机会接近全局最优。因此，可以得出具有选择操作的 QPSO 算法将会改善 QPSO 的全局搜索能力。

3.4.2　采用锦标赛选择操作的 QPSO 算法(QPSO-TS)

在锦标赛选择法中，随机地从种群中挑选一定数目的个体，称为竞赛规模，然后将最好的个体选作父个体，这个过程重复进行完成个体的选择。在 QPSO-TS 中，竞赛规模为 1，即每次随机从种群中选择一个个体 k，计算其目标函数值，并与当前粒子的目标函数值相比较，如果好于当前粒子的目标函数值，则 p_i 由 P_k 和 P_i 决定，否则， p_i 由 P_g 和 P_i 决定。这个选择操作的伪代码描述如下：

```
Selection_T:
从粒子群中随机选择一个粒子 k
    if f(Pk) < f(Pi)
    T=Pk;
    else
    T=Pg;
    endif
    Return T
```

通过上面的选择操作选出 T 后， p_i 的坐标为

$$p_{i,j}(t)=\phi\bullet P_{i,j}(t)+(1-\phi)\bullet T_j(t)$$

QPSO-TS 的算法过程描述如下：

```
QPSO_TS:
在问题空间中随机初始化粒子群中粒子的位置：Xi=X[i][:];
初始化粒子的个体最优位置：Pi=Xi;
while the stop criterion is not met do
Compute the mean best position C[:] by equation (2.44);
for i = 1 to swarm size M
If f(Xi)<f(Pi) then Pi=Xi;    Endif
Find the Pg=arg min f(P[g][:]);
    T=Selection_T;
for j=1 to D
   φ=rand(0,1);   u=rand(0,1);
            p[i][j]= φ*P[i][j]+(1−φ)*T[j];
if (rand(0,1)>0.5)
    X[i][j]=p[i][j]+α*abs(C[j]−X[i][j])*ln(1/u);
Else
    X[i][j]= p[i][j]−α  *abs(C[j]−X[i][j])*ln(1/u);
Endif
Endfor
Endfor
Endwhile
```

3.4.3 采用轮盘赌选择操作的 QPSO 算法(QPSO-RS)

轮盘赌选择法是遗传算法中一种常用的选择策略。它是将集合元素映射到一个区间，将此区间按情况分成若干个子段(每段对应一个元素)。然后产生 n 个有效随机数(即都落在区间内)，统计随机数落在各个子段的频率，频率最高的子段对应的元素即为被选中元素。在 QPSO-RS 算法中，由轮盘赌选择操作选中群体中的一个个体 k，则 p_i 由 P_k 和 P_i 决定。轮盘赌选择操作的伪代码如下：

```
Selection_R:
    n 为种群大小;
```

```
r 为 n 个有效随机数(即都落在区间内);
for i=1:n
    sum_P = 0;
    j = ceil(m*rand);    %产生 1～m 之间的随机整数
    while sumP < r(i)
      sum_P = sum_P + P(mod(j-1,m)+1);
      j = j+1;
    end
   R(i) = mod(j-2,m)+1;
end
Return R
```

通过上面的选择操作选出 R 后，p_i 的坐标为

$$p_{i,j}(t) = \phi \bullet P_{i,j}(t) + (1-\phi) \bullet R_j(t)$$

QPSO-RS 的算法流程类似于 QPSO-TS 的算法流程，只是吸引点 p_i 的方程不同。

QPSO-TS 和 QPSO-RS 参数值的设定与 QPSO 算法一致。

3.4.4　算法的收敛性分析

1. 全局收敛性准则

这里从 Solis 和 Wets 提出的收敛准则着手进行基于选择操作的 QPSO 的收敛性分析。下面首先定义目标函数 $f(x)$ 的一个最优化区域：

$$R_\varepsilon = \{z \in S \mid f(z) < \phi + \varepsilon\} \tag{3.65}$$

式中，$\varepsilon > 0$；S 是 $f(x)$的解空间；ϕ 是 $f(x)$的最优解。

假如算法在最优化区域找到了一点，那么这个点就是被找到的函数全局最小值的一个可接受的近似值。

现在需要考虑一个算法是否确实能够在搜索空间中到达最优化区域中。对于最小化问题，如果算法满足下面的假设，那么这个算法可能是全局收敛的。

假设 3.4：对于 S 的任意 Borel 子集 A，若其测度 $v[A]>0$，则有

$$\prod_{k=0}^{\infty}(1-\mu_k[A])=0 \tag{3.66}$$

式中，$\mu_k[A]$ 是由测度 μ_k 所得到的 A 的概率。

这就意味着对于测度大于 0 的任意一个 S 中的集合 A 来说，如果采用随机取样的方法，那么它无穷多次重复错过集合 A 的概率必定为 0。由于 $R_\varepsilon \subset S$，暗示了所有在最优化区域的样本点的概率肯定是非零值。由 Solis 和 Wets 准则得出下面的定理：

定理 3.6：假定算法在每一步都有最优解，并且目标函数 f 是可测函数；区域 S 是 R^D 的可测子集，并且满足假设 3.4。设 $\{z_k\}_{k=0}^{\infty}$ 为算法生成的解序列，可得

$$\lim_{k\to+\infty} P[z_k \in R_\varepsilon]=1 \tag{3.67}$$

式中，$P[z_k \in R_\varepsilon]$ 是第 k 步算法生成的解 $z_k \in R_\varepsilon$ 的概率。

2. 基于选择操作的 QPSO 算法的全局收敛性

下面的证明将基于选择操作的 QPSO 算法置于全局随机搜索算法的框架中研究，以定理 3.6 的结论进行证明。由于 QPSO-TS 和 QPSO-RS 每一次迭代都存储了全局最优的位置，所以基于选择操作的 QPSO 算法似乎满足假设 3.4。这里使用下标 k 取代迭代次数 t，从而 $x_{i,j,k}$ 代表了第 i 个粒子的第 j 维在第 k 次迭代中的位置向量，在下面的分析中其他的变量用相同的方式来表示。

引理 3.3：基于选择操作的 QPSO 算法满足假设 3.4。

证明：在任意一次迭代步 k，$p_{i,j,k}$ 都被视为一个随机变量，第 i 个粒子的第 j 维在 k+1 步的条件概率密度函数为

$$Q(x_{i,j,k} \mid p_{i,j,k})=(1/L_{i,j,k})\exp(-2\left|x_{i,j,k}-p_{i,j,k}\right|/L_{i,j,k}) \tag{3.68}$$

式中，$L_{i,j,k}=\alpha\left|C_{j,k}-x_{i,j,k}\right|$。

因此可以得到粒子 i 的第 j 维在 k+1 步的概率密度函数为

$$Q(x_{i,j,k+1})=\int_{-\infty}^{+\infty}Q(x_{i,j,k}\mid p_{i,j,k})Q(p_{i,j,k})\mathrm{d}p_{i,j,k} \tag{3.69}$$

式中，$Q(p_{i,j,k})$ 是吸引点 $p_{i,j,k}$ 的边缘概率密度函数，满足

$$Q(p_{i,j,k})\sim U(a_{i,j,k},b_{i,j,k}) \tag{3.70}$$

式中，$a_{i,j,k}=\min(P_{i,j,k},(P_g)_{j,k})$，$b_{i,j,k}=\max(P_{i,j,k},(P_g)_{j,k})$。

因为 $0<\sup L_{i,j,k}<\infty$，从 $M(x_{i,j,k+1}\mid p_{i,j,k})=\mathbf{R}$ 是 $Q(x_{i,j,k+1}\mid p_{i,j,k})$ 的支撑，可以得到 $Q(x_{i,j,k+1})$ 的支撑为

$$M(x_{i,j,k+1})=M(x_{i,j,k+1}\mid p_{i,j,k})\cup M(p_{i,j,k})=\mathbf{R}\cup[a_{i,j,k},b_{i,j,k}]=\mathbf{R} \tag{3.71}$$

其中，根据式(3.70)，$M(p_{i,j,k})=[a_{i,j,k},b_{i,j,k}]$。从而，在每一步 $k+1(k\geqslant 0)$，$M(x_{i,j,k+1})=\mathbf{R}$。

因为粒子 i 的概率密度函数可以表示为

$$Q(x_{i,k+1})=\prod_{j=1}^{n}Q(x_{i,j,k+1}) \tag{3.72}$$

所以 $Q(x_{i,k+1})$ 的支撑可以表示为

$$M(x_{i,k+1})=\prod\nolimits_{j=1}^{n}M(x_{i,j,k+1})=\underbrace{\mathbf{R}\times\mathbf{R}\cdots\mathbf{R}}_{n}=\mathbf{R}^n\supset S \tag{3.73}$$

定义 $\mu_{i,k+1}$ 为上面分布 $Q(x_{i,k+1})$ 的概率测度，因此对于 S 的任意 Borel 子集 p，当满足 $v[p]>0$，可以得到

$$\mu_{i,k+1}[p]=\int_A Q(x_{i,k+1})\mathrm{d}x_{i,k+1}=\int_A\prod_{j=1}^{n}Q(x_{i,j,k+1})\mathrm{d}x_{i,k+1} \tag{3.74}$$

因为 $\mathrm{p}\subset M(x_{i,j,k+1})$，对于所有的 $k\geqslant 0$，$Q(x_{i,k+1})$ 是可积分的。因此，从式(3.74)，则必定有

$$0<\mu_{i,k+1}[p]<1 \tag{3.75}$$

对于所有 $k\geqslant 0$，可以得到所有粒子支撑的并集为

$$M_{k+1}=\bigcup\nolimits_{i=1}^{m}M(x_{i,k+1})=\mathbf{R}^n\supset S \tag{3.76}$$

式中，M_{k+1} 是分布 μ_{k+1} 在样本空间的支撑。

由 μ_{k+1} 产生的对 p 的概率测度可以由下式计算得到

$$\mu_{k+1}[p]=1-\prod_{i=1}^{m}(1-\mu_{i,k+1}[p]) \tag{3.77}$$

式中，m 为粒子的个数。

从不等式(3.75)和等式(3.77)中，可以得到：对于 $k=0,1,2,\cdots$，有 $0<\mu_{k+1}[p]<1$。在第 0 步，在一个有界的区域 M_0 内初始化粒子，其中 $M_0\subset\mathbf{R}^n$。如果 $p\cap M_0\neq\phi$，则 $0<\mu_0[p]<1$，否则 $\mu_0[p]=0$。

总之，可以得到下面的结果：

$$\begin{aligned}\prod_{k=0}^{\infty}(1-\mu_k[p])&=(1-\mu_0[p])\prod_{k=0}^{\infty}(1-\mu_{k+1}[p])\\&=(1-\mu_0[p])\bullet 0=0\end{aligned} \tag{3.78}$$

从式(3.76)可以得到基于选择操作的 QPSO 算法满足假设 3.4。

定理 3.7：基于选择操作的 QPSO 算法是一种全局收敛的算法

证明：因为基于选择操作的 QPSO 算法满足假设 3.4，由定理 3.6，可以得到基于选择操作的 QPSO 算法是一种全局收敛的算法。

3.5　本章小结

本章详细介绍了遗传算法、粒子群优化算法和量子粒子群优化算法的优化过程及收敛性分析，为进行多序列比对提供理论基础。

参 考 文 献

[1] Thomsen R, Boomsma W. Multiple sequence alignment using SAGA: investigating the effects of operator scheduling, population seeding,

and crossover operators[J]. Appl.Evol.Comput., 2004(3005): 113-122.

[2] Gondro C, Kinghorn BP. A simple genetic algorithm for multiple sequence alignment[J]. Genetics and Molecular Research, 2007, 6(4): 964-982.

[3] Fernando José Mateus da Silva, Juan Manuel Sánchez Pérez, Juan Antonio Gómez Pulido, etal. Optimizing Multiple Sequence Alignment by Improving Mutation Operators of a Genetic Algorithm[J]. 2009 Ninth InternationalConference on Intelligent Systems Design and Applications. Pisa: IEEE, 2009,1257-1262.

[4] 胡桂武，郑启伦，彭宏. 基于遗传算法与星比对的多序列比对混合算法[J]. 计算机应用，2004，24(5)：90-92.

[5] 张维梅. 基于遗传算法和蚁群算法的多重序列比对[J]. 潍坊学院学报，2008，8(4)：88-90.

[6] 司秀华，陈国良. 一种多搜索策略的多生物序列比对自适应遗传算法[J]. 小型微型计算机系统，2006，27(5)：854-857.

[7] Zne-Jung Lee, Shun-Feng Su, Chen-Chia Chuang, etal. Genetic algorithm with ant colony optimization(GA-ACO) for multiple sequence alignment[J]. Applied Soft Computing, 2008, 8(1): 55-78.

[8] Y.P. Chen, W.C. Peng, M.C. Jian, “Particle swarm optimization with recombination and dynamic linkage discovery,” IEEE Trans. Syst. Man Cybernet. Part B, vol. 37, no. 6, 2007, pp. 1460-1470.

[9] M. Pant, T. Radha, V.P. Singh, “A new particle swarm optimization with quadratic interpolation,” in: Proceedings of IEEE International Conference on Computational Intelligence and Multimedia Applications, 2007, pp. 55-60.

[10] P.J. Angeline, “Using selection to improve particle swarm optimization,” in: Proceedings of IEEE International Conference on Evolutionary Computation, 1998, pp. 84-89.

[11] Z.-H. Zhan, J. Zhang, Y. Li, H.H. Chung, “Adaptive particle

swarm optimization," IEEE Trans. Syst. Man Cybernet. Part B,vol. 39, no.6, 2009, pp: 1362-1381.

[12] Y.H. Shinn, S.L. Hung, H.L. Weei, J.H. Shinn, "Orthogonal particle swarm optimization and its application to task assignment problems,"IEEE Trans. Syst. Man Cybernet. Part A, vol. 38, no. 2, 2008, pp. 288-298.

[13] Z.-H. Zhan, J. Zhang, Y. Li, Y.-H.Shi, "Orthogonal learning particle swarm optimization," IEEE Trans. Evol.Comput., vol. 15, no. 6, 2011, pp. 832-847.

[14] B. Alatas, E. Akin, O. Bedri, "Chaos embedded particle optimization algorithms," Chaos Solitons Fractals, vol. 40, no. 4, 2008, pp. 1715-1734.

[15] J. Sun, X. J. Wu, V. Palade, et al., "Convergence analysis and improvements of quantum-behaved particle swarm optimization," Information Sciences, vol. 193, 2012, pp. 81-103.

中　篇

多序列比对模拟篇

第 4 章　遗传算法在多序列比对中的应用

4.1　基本遗传算法模拟多序列比对

4.1.1　引言

随着人类基因组计划的实施和信息技术的发展，生物序列数据呈爆炸式增长，如何有效分析和处理这些大型序列数据(即序列分析)成为生物信息学的首要任务。序列比对是生物序列分析的主要方法，也是生物信息学中具有挑战性的问题之一，其在序列装配、序列注释、蛋白质的结构和功能预测、系统发育和进化分析等方面均有广泛应用。根据参与比对的序列数目，序列比对分为双序列比对和多序列比对。张春霆、郭茂祖和 Chuong 等总结了国内外近年来序列比对的研究进展，认为双序列比对已有较成熟的动态规划算法，但是多序列比对在目前还缺乏快速而有效的算法。

多序列比对是一个 NP 完全的组合优化问题，解决此问题的传统算法是渐进算法或迭代算法，但是随着序列长度和条数的增多，时空复杂性急剧上升，设计一个具有高敏感、高精度且低复杂度的算法成为解决生物多序列比对的瓶颈问题。Chuong 等在文中引用 258 个文献对多序列比对做了非常全面的综述，同时也分析并预测了未来该问题的研究方向：虽然多序列比对问题已经研究了几十年，但是多序列比对的研究一直蓬勃发展，每一年都有数十个描述多序

列比对新方法的文章发表。最近的许多研究都表明，虽然在提高比对的精度和比对处理范围可扩展性方面都取得了相当大的进展，但是在该方面仍有很大的发展空间值得研究和完善。

Wang 等已经证明：基于 SP 度量的多序列比对是一个 NP 问题。实际上，除了个数较少序列较短的比对问题外，多序列比对基本上都是采用启发式算法。目前国际上最具代表性的启发式算法有两大类：渐进比对和迭代比对。由 Thompson 和 Notredame 等开发的 CLUSTALW 和 T-Coffee 是基于渐进算法最常用的程序软件包，这种算法的比对速度很快，但其依赖于指导树的构建，敏感性较差。迭代比对是另一类有效的多序列比对策略，它基于一个能产生比对的算法，并通过迭代方式精细多序列比对，直到比对结果不再改进为止。这类算法不能提供获得优化比对结果的保证，速度也不能和渐进算法相比，但却具有鲁棒性和对比对序列个数不敏感等特性。近年来，迭代算法也被越来越多地应用到序列比对中去，如遗传算法、蚁群算法、隐马尔科夫等。Notredame 等首先提出用遗传算法来解决多序列比对问题；梁艳春等在隐马尔科夫(HMM)模型中结合了粒子群优化和模拟退火进行学习，并且在训练过程中结合人工免疫策略，在运行时间和 SP 值等参数上得到了较好的效果；廖波等将蚁群算法和渐进算法结合起来，解决多序列比对问题；Silva 等提出了将局部最优搜索融入遗传算法中的新算法，提高了算法的准确度；还有其他一些以遗传算法为主的序列比对算法；邹权和郭茂祖等综述了常见的启发式方法，除了遗传算法和粒子群优化算法，还有许多其他的启发式方法，尽管在应用到多序列比对问题上作了许多尝试，但中间还存在着一些十分难处理的问题，因而还没有形成基于这些方法的主流软件。张鑫源等根据算法性能分别比较了遗传算法与粒子群优化算法，认为相对于粒子群优化算法，遗传算法速度较慢，但不容易陷入局部最优解，所以本章基于遗传算法研究多序列比对问题。

4.1.2 多序列比对问题及数学描述

一条长度为 L 的序列是 L 个字符组成的字符串，字符取自于字母表{A, V, L, I, F, P, M, S, T, C, W, Y, N, Q, D, E, K, R, H, G}，分别代表蛋白质的 20 个氨基酸残基类型。对于蛋白质序列，给定包含 N 个序列的序列集 $S=\{S_1,S_2,\cdots,S_N\}$，$N\geqslant 2$， $S_i=S_{i1}S_{i2}\cdots S_{il_i}(1\leqslant i\leqslant N)$，$S_{ij}\in\sum(1\leqslant j\leqslant l_i)$，$l_i$ 是第 i 条序列的长度，则一个序列比对可定义为一个矩阵 $\boldsymbol{A}=(a_{ij})$，其中 $1\leqslant i\leqslant N$，$1\leqslant j\leqslant l, \max(l_i)\leqslant l\leqslant\sum_{i=1}^{N}l_i$ 。矩阵必须满足下列三个条件：

(1) $a_{ij}\in\sum\cup\{-\}$，其中“－”代表空位。

(2) 矩阵中的第 i 行去掉“－”后，即得到字符串 S_i 。

(3) 矩阵中不包含字符全是空格的列。

4.1.3 算法设计

1. 编码

根据 1.3.3 节，多序列比对可以看作一个矩阵(或二维表)，每一行代表一个序列，每一列代表一个残基的位置。将序列依照下列规则填入表中：

(1) 一个序列所有残基的相对位置保持不变。

(2) 通过插入空位，将不同序列间相同或相似的残基调整至同一列，即尽可能将序列间相同或相似残基上下对齐。

(3) 插入空位后各序列的长度相等。将 m 条序列比对结果存放在矩阵(或二维表) $\boldsymbol{A}_{m\times L}$ 中，其中 m 是参加比对的序列数，L 是插入空位后的序列长度(简称比对长度)。

表 4.1 表示 6 个短序列的比对结果，此例中 m=6，L=14。

表 4.1　多序列比对示例

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
I	Y	Y	D	G	G	A	V	–	E	A	L	C	A	M
II	Y	D	D	–	G	A	L	V	E	A	L	C	A	M
III	F	D	E	G	G	–	L	V	Q	A	–	C	F	–
IV	F	Y	E	G	G	I	V	V	Q	A	V	–	–	M
V	Y	D	D	G	G	I	L	V	–	–	L	C	A	N
VI	Y	Y	E	–	G	A	–	V	Q	A	V	C	E	M

2. 适应度函数

适应度函数是遗传算法与应用问题的唯一接口，这里采用最流行的 SP 作为衡量标准。SP 函数为

$$f(S)=\sum_{h=1}^{L}\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}\boldsymbol{Cost}(S_{ih},S_{jh}) \tag{4.1}$$

式中，$Cost(*, *)$是 $\boldsymbol{A}^*\times\boldsymbol{A}^*\rightarrow\mathbf{R}$ 为二元实值函数，称式(4.1)的值为比对 S 的分值。

适应度值越高说明序列比对效果越好。$\boldsymbol{A}^*$中的每个元素中矩阵 $\boldsymbol{Cost}$ 的具体取值为

$$\boldsymbol{Cost}(S_{ih},S_{jh})=\begin{cases}S_{aa}=Score(a,a) & \text{如果两个非空位残基相同(匹配)}\\ S_{ab}=Score(a,b) & \text{如果两个非空位残基不同(不匹配)}\\ S_{a-}=Score(a,-)=0 & \text{残基与空位的分数为0(空位罚分另计)}\end{cases} \tag{4.2}$$

式中，L 为比对序列长度；m 为参加比对的序列条数；S_{ih} 为第 i 条序列第 h 个残基，匹配和不匹配的分数通常由计分替换矩阵给出。

3. 遗传算子的设计

1) 选择算子

算法采用了两种选择方式：精英保留法(最佳个体保存法)、轮

盘赌选择法，对个体的适应度值按大小排序，将前 10%的个体保留到下一代，其余 90%的个体采用轮盘赌选择法选取父体。

2) 交叉算子

采用单点纵向交叉方式，在种群中随机配对，根据交叉概率选定某对进行交叉，在父代个体 1 中随机设定一个交叉点，在父代个体 2 中找到相应的交叉点，实行交叉时，该点前或后的两个个体的部分结构进行交换，并生成两个新个体。例如图例，随机选择交叉点 2，进行单点纵向交叉的过程如图 4.1 所示。

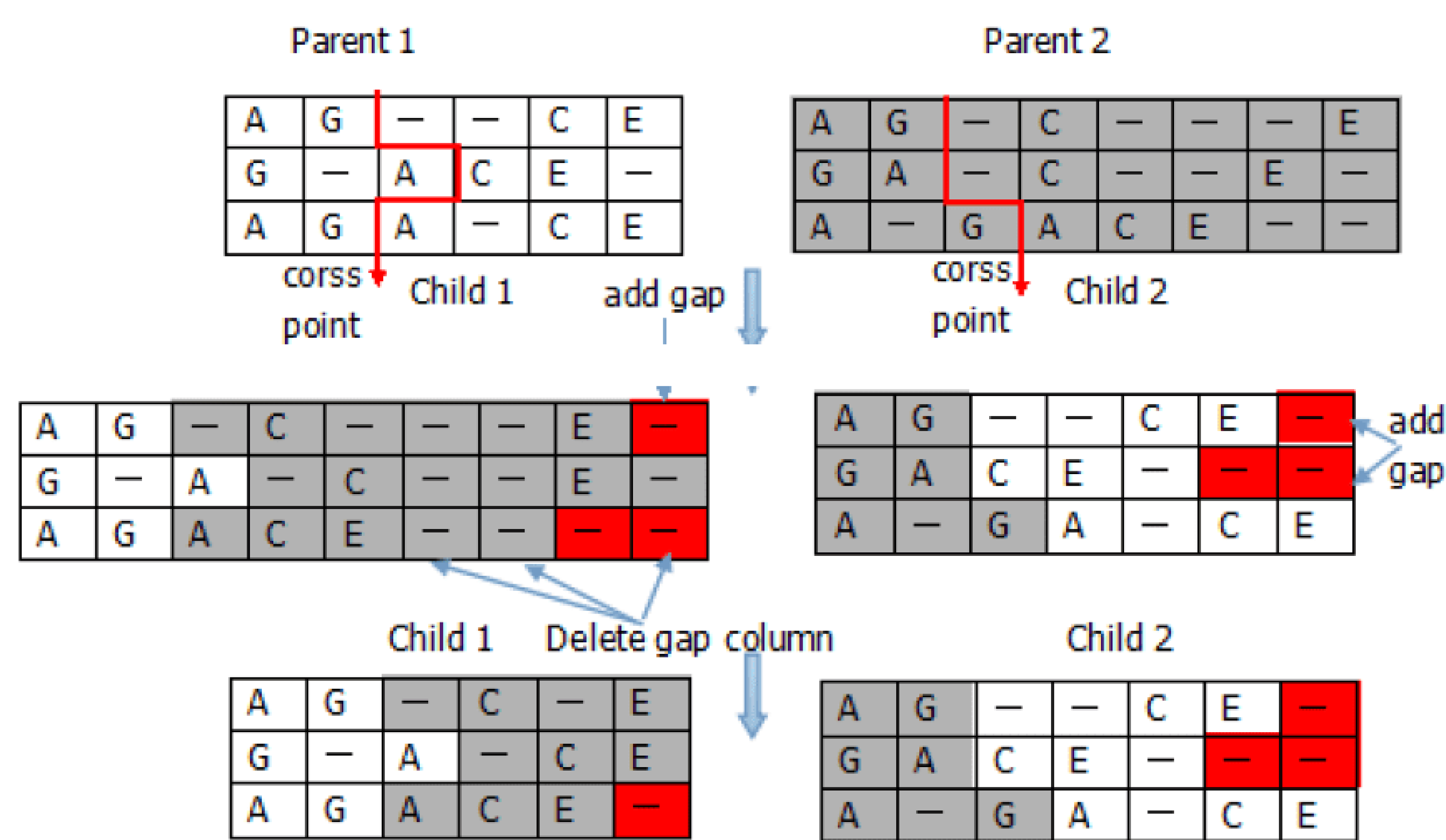


图 4.1　单点纵向交叉的过程

3) 变异算子

使用一点变异法：对于多序列比对问题，插入码是“—”，对于群体中的个体按照变异概率随机选择一个个体作为父本，随机在某条序列中选择一个位置，如果这个位置上的字符为空格，则随机另选一个非空格位置，将空格移动到这个位置上，其他的字符向左或向右移动。根据保优原则，如果新产生的个体的适应度函数值大于原个体的适应度数值，则代替原个体，否则保持原个体。

4. 流程图

基本遗传算法的流程如图 4.2 所示。

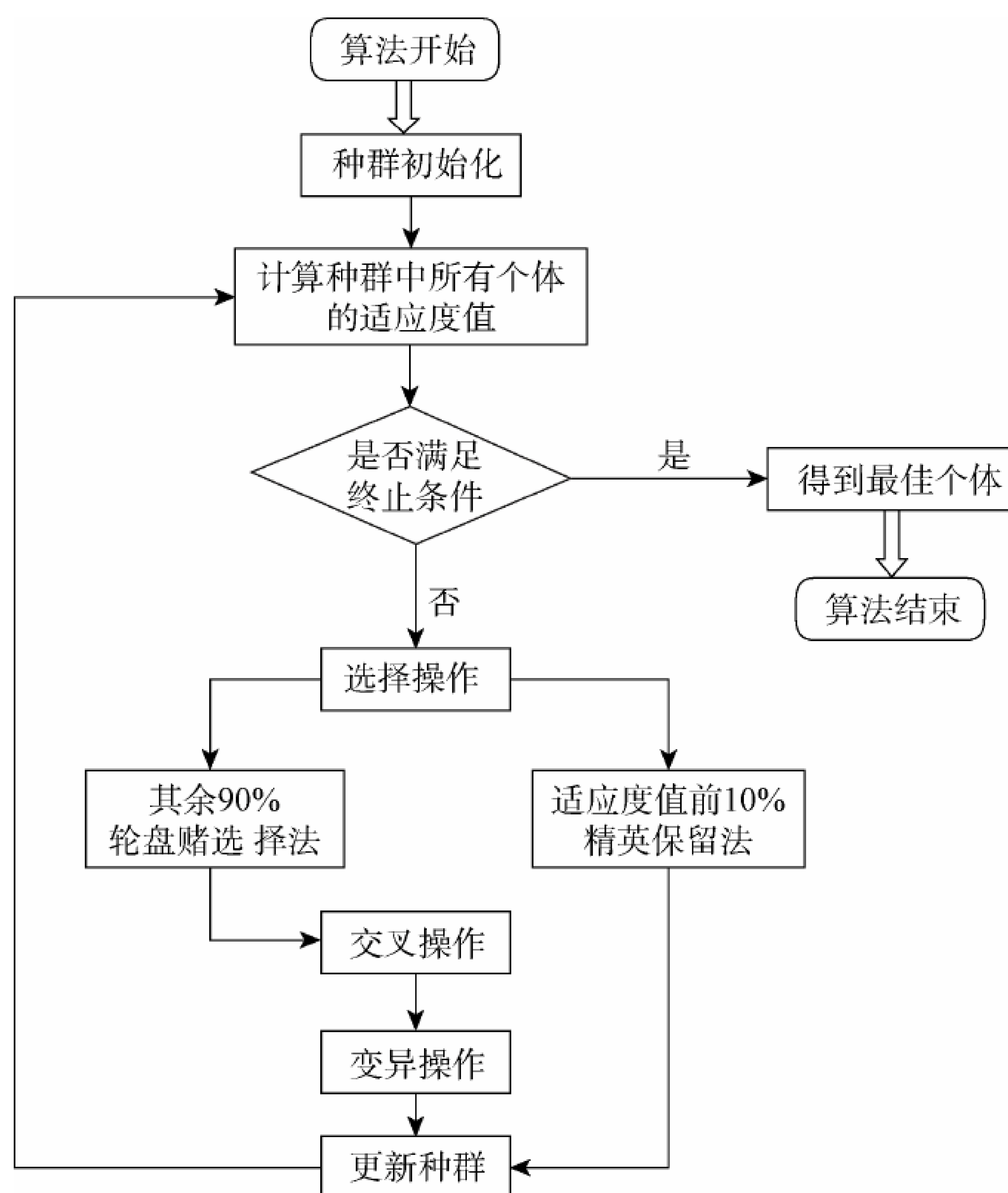


图 4.2　基本遗传算法流程图

4.1.4　实验算例与分析

1. 实验结果的评价标准

为了对数据进行统一且精确的比较，本章选用多序列比对基准数据库 BAliBASE2.0 数据库中提供的参考比对作为评价算法的测试数据对象，并与现有的在线比对工具的结果进行比较研究。BAliBASE 提供了两个评价分值 SPS 和 CS，分别用于评价与 BAliBASE 中参考比对进行比较的一个测试比对的质量，代表着正确识别出保守位点的百分比，所以 SPS 和 CS 越高，说明比对的结果越接近于参考序列，比对效果越好。

第一个函数是 Sum-of-Pairs Score(SPS)：设有 N 个比对测试序列，构成 M 列，标记第 i 列的比对列为 $A_{i1},A_{i2},\cdots,A_{iN}$。对每一对残

基 A_{ij} 和 A_{ik}，定义变量 p_{ijk}，如果 A_{ij} 和 A_{ik} 在比对的参考结果中也位于同一列，则 $p_{ijk}=1$，否则 $p_{ijk}=0$。定义变量 S_i 为第 i 列的得分：

$$S_i = \sum_{j=1, j\neq k}^{N} \sum_{k=1}^{N} p_{ijk} \tag{4.3}$$

记 SPS 为最终比对结果的得分，则

$$SPS = \sum_{i=1}^{M} S_i \Big/ \sum_{i=1}^{M_r} S_{ri} \tag{4.4}$$

式中，M_r 是参考比对结果中的列数；S_{ri} 是参考比对结果中第 i 列的得分。

用该函数进行评价时，比对结果的得分越多，说明比对的结果越好。

第二个函数是 Column Score(CS)：对比对结果的第 i 列，如果这列中所有的残基在参考比对中也位于同一列，则 $C_i=1$，否则 $C_i=0$。按照下面的公式对比对结果进行 CS 打分。

$$CS = \sum_{i=1}^{M} C_i / M_r \tag{4.5}$$

式中，M_r 是参考比对结果中比对列的个数。

2. 实验参数

应用遗传算法模拟多序列比对的实验参数汇总见表 4.2。

表 4.2　实验参数

Parameter	451c_ref1	Fitness function	Sum-of-pairs
Population size	50	Score matrix	BLOSUM62
Generations	2000	Gap open penalty	−3
Selection strategy	Roulette wheel	Gap extend penalty	−0.15
Elitist rate	0.1	Crossover probability	0.6
Crossover operator	One point	Mutation probability	0.2
Mutation operator	One bit	Coding	Two-dimensional
Programming Language	MATLAB		

3. 实验结果与分析

451c_ref1 原参考序列如图 4.3 所示。

```
mseq =

ATPAELATKA_GCAVCHQPTAKGLGPSYQEIAKKYKGQAGAPALMAE__RVRKGSVGIFG______KLPMTPTPARPISDADLKLVIDWIL___
ASPEEIY_KA_NCIACHGENYE____GVSGPSLKGVGDKKDVAEIKT__KIEKGG____________NGMPSGL___VPADKLDDMAEWVSKI_
ADGAALY_KS__CIGCHGADGS___KAAMGSAKPVKGQ_GAEELYK___KMKGYADGSYG___GERKAMMTNAV_KKYSDEELKALADYMSKL_
DAEAGQG_KVAVCGACHGVDGN____SPAPNFPKLAGQ_GERYLLKQLQDIKAGSTPGAPEGVGRKVLEMTGML_DPLSDQDLEDIAAYFSSQK
QDGEALF_KSKPCAACHSVDTKMVGPALKEVAAKNAGVEGAADTLAL__HIKNGSQGVWG______PIPMPPNP___VTEEEAKILAEWVLSLK
```

图 4.3　451c_ref1 原参考序列

模拟出来的 451c_ref1 序列的比对结果见图 4.4。

```
ans =

ATPAELATKA_GCAVCHQPTAKGL_GPSYQEIAKKYK__GQAGAPAL_MAER_VRKG_SVGIFG_KLPMTPTP__ARPISDAD_L_KLVIDW_IL_
ASPEEI_YKAN_CIACHGENYEGV_SGP____SLKGV__GDKKDVAEI_KTK_IE__K____GGN_GMPSGL__VPA_DKLDD_MAEWVS_KI__
ADGAAL_YKS__CIGCHGADGSKAAMGSAKPVKG_QGAEE_LY_KK_M_KGYA_D_GS__YGGERKAMMTNAVKKYSD_EELKA_LADYM_SKL__
DAEAGQ_GKVAVCGACHGVDGN_SPAPNFPKLAG_QGER_YLLKQLQDIKAGSTPGAPEGVG_RKVLEMTGMLDPLSDQ_DLED_IAAYFS_SQ_K
QDGEAL_FKSKPCAACHSVDTKMV_GPALKEVAAKN_AG_VEGAAD_TLALH_IKNG_SQGVW_GPIPMPPNP_VT_E_EEAKI_LAEWVL_SL_K
       *   *  **                                                     *
```

图 4.4　451c_ref1 序列的比对结果图(*列是全对齐列)

其他的模拟结果数据统计见图 4.5。

```
print_cm06_2st =

    besttseq: {[6x96 char]  [6x96 char]  [6x89 char]  [6x96 char]  [6x95 char]  [6x95 char]  [6x95 char]  [6x93 char]}
      seqfit: {1x8 cell}
    num_iter: [2000 2000 2000 2000 2000 2000 2000 2000]
         ksx: [4 2 3 5 3 5 2 3]
     bestsps: [0.6836 0.6425 0.6466 0.6068 0.6562 0.6521 0.7808 0.6438]
     besttcs: [0.3333 0.3438 0.3258 0.2708 0.3474 0.3053 0.4211 0.3011]
        time: [2.6931e+003 2.6500e+003 2.6971e+003 2.6402e+003 2.6532e+003 2.6265e+003 2.6737e+003 2.6558e+003]
```

图 4.5　模拟结果数据统计

451c_ref1 的 SPS 值与迭代次数的关系图见图 4.6。

应用遗传算法比对的结果与当前热门在线测试软件比对结果的比较见表 4.3。

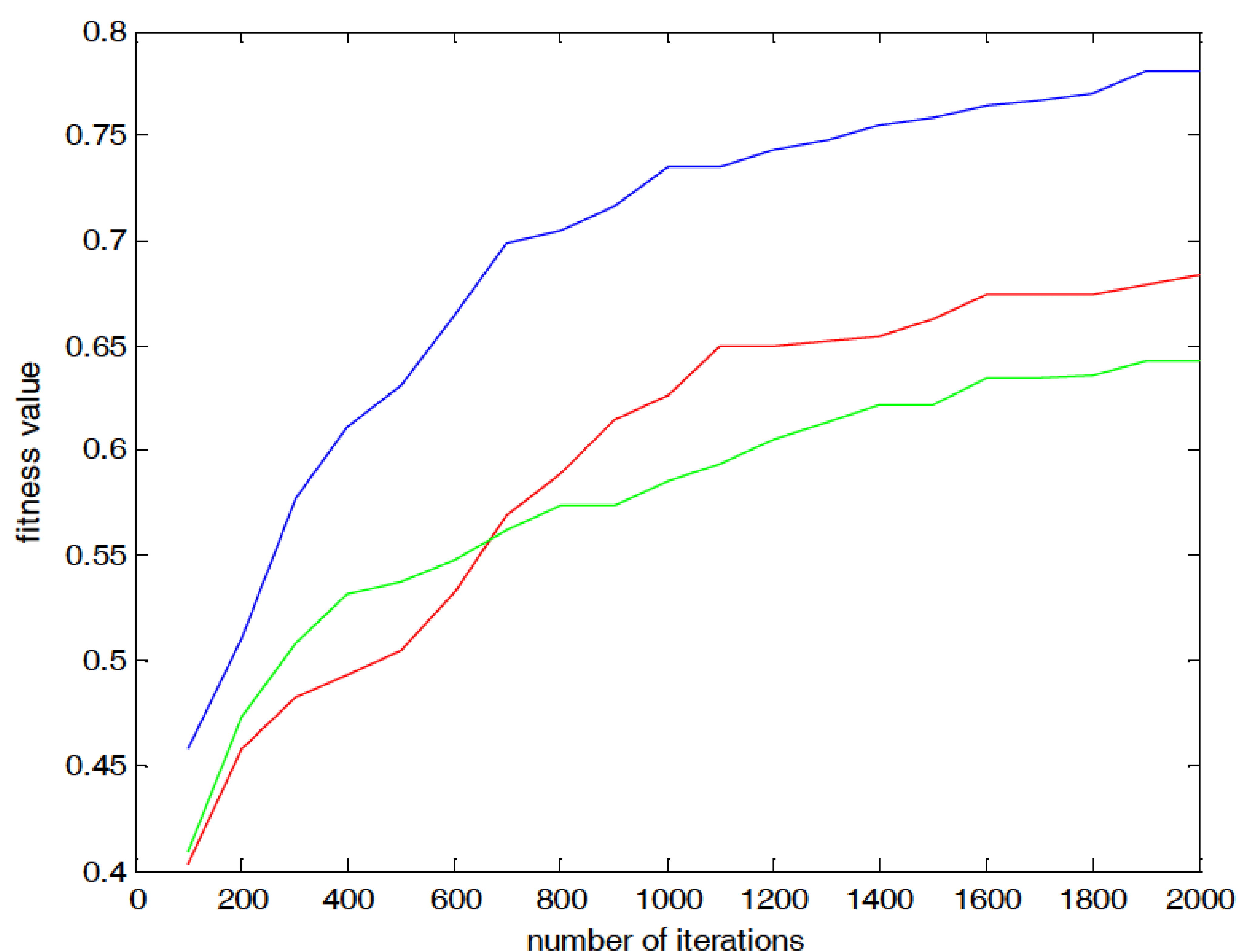


图 4.6　451c_ref1 的 SPS 值与迭代次数的关系图

表 4.3　遗传算法与其他算法的结果比较

方法	SPS	CS	SIZE	完全匹配列*	个数	耗时(s)
参考序列	1	1	5×94	KCCHGM	6	
Clustalw2	0.5438	0.3297	5×91	CCHG	4	<10
Clustal omega	0.5342	0.4066	5×91	CCHG	4	<10
Kalign	0.5726	0.2708	5×96	CCHG	4	<10
MAFFT	0.5973	0.3125	5×96	CCHG	4	<10
MUSCLE	0.5699	0.30	5×90	CCHG	4	<10
T-Coffee	0.6178	0.3626	5×91	CCHG	4	<10
bpso	0.7233	0.4063	5×96	CCH	3	1386
Genetic	0.7808	0.4211	5×95	KCCHM	5	2650

4.1.5　结论

比对序列 451c_ref1，与其他算法相比较，应用基本遗传算法

迭代 2000 次可以得到较为满意的结果，完全匹配列最为接近参考序列。

虽然比对结果较为理想，但是也存在一些问题：

(1) SPS 最高，但 CS 不是最高。

(2) 匹配列最多，分数不是最高。

(3) 耗时太长。

(4) 只测试了一个短序列，结果与在线软件相比更好。长序列(>200)结果不太理想。

(5) 从收敛曲线看，迭代 2000 次仍未收敛，迭代次数增加，结果更好，耗时更长。

因此，为了更好地提高遗传算法的比对效率，4.2 节和 4.3 节将分别从初始种群和交叉算子两方面进行优化，从而达到改进遗传算法的目的。

4.2　改进遗传算法之初始种群优化

4.2.1　引言

初始种群是遗传算法中极为关键的部分，Gondro 认为初始种群的质量直接影响到遗传算法的收敛速度，适应度值高的种群能够很快地收敛到接近最优解的解，因此，一个高品质的初始种群是改进遗传算法的关键所在。生成初始种群个体的传统做法是在序列中随机插入空位，这样的个体没有任何生物学的比对意义，难免会产生一些适应度值较低的个体，造成计算成本昂贵。如果在序列中插入空位时考虑多序列比对的生物特性，这样生成的个体所组成的初始种群符合多序列比对的生物特性，具有较高的适应度值，可从个体角度上提高初始种群的质量。

多序列比对有很多常见的在线测试工具，如 MAFFT、ClustalW、

MUSCLE 等，应用这些在线工具能快速地得到较好的比对结果。但是这些结果对各参数的依赖性很强，不同参数下得到的结果很不一样。而目前尚没有有效的方法直接确定最优参数值，故很难直接通过在线工具得到局部最优解。这些在线比对结果虽不能作为最优解，但可以作为多序列比对的参考。如果将这些优质比对结果作为初始种群的一部分个体，经过遗传操作和迭代后将能得到更好的比对结果。根据在线测试工具快速比对的特点，从上述在线测试工具中选择参数简单且比对效果较好的 MAFFT 比对工具，将其比对结果作为优质种子，以一定比例加入到初始种群中，可从整体角度上提高初始种群的质量。

经过这两个优化过程，可以生成更高质量的初始种群。在数值模拟中，从 BAliBASE2.0 数据库随机选择序列簇作为实验对象，以 SPS 值作为评估比对质量的标准，通过实验验证，这两个做法的组合优化了初始种群的质量，提高了多序列比对的计算效率，从而达到改进的目的。

4.2.2　优化原理

1. 插入连续空位

在应用遗传算法进行多序列比对时，首先要生成初始种群。一般做法是在序列中随机插入若干的空位，插入空位后的序列长度不超过最长序列长度的 1.2 倍，据此定义空位率为 0.2。众多文献的初始种群个体都是在序列中随机插入空位，且每个个体的维数(插入空位后的序列长度)都是最长序列长度的 1.2 倍，即同维个体，这种应用传统做法生成的个体，这里称之为随机插入空位固定长度的个体，简称 randfix。

但是对于多序列比对问题，从生物进化的角度认为：在非空位位点间加入新的空位不如在空位位点插入具有连续较长的空位更能体现进化[13-14]。根据手工比对好的参考序列或在线测试软件比对结果，能观察出其空位分布是连续的。例如 BAliBASE2.0 中 1csp_ref1，

其手工比对好的参考序列见图 4.7。

```
---MLEGKVKWFNSEKGFGFIEV-EGQDDVFVHFSAIQG----EGFKTLEEGQAVSFEIVEGNRG-PQAANVTKEA
MSGKMTGIVKWFNADKGFGFITPDDGSKDVFVHFSAIQN----DGYKSLDEGQKVSFTIESGAKG-PAAGNVTSL-
---MATGTVKWFNAEKGFGFIAQDGGGPDVFVHYSAINA----TGFRSLEENQVVNFDVTHG-EG-PQAENVSPA-
-----KGTVKWFSDQKGFGFITPDDGGEDLFVHQSGIRS----EGFRSLAEGETVEFEVESGGDGRTKAVDVTGP-
----VLGTVKWFNVRNGYGFINRNDTKEDVFVHQTAIKKNNPRKYLRSVGDGETVEFDVVEGEKG-AEAANVTGP-
```

图 4.7　1csp_ref1 参考序列

所以在序列中插入这样的连续空位更符合其生物特性，这里称之为具有生物特性的固定长度的个体，简称 biofix。

众多文献的初始种群个体维数都是最长序列长度的 1.2 倍，也就是上面提到的同维个体，即固定长度的个体。张琎等从具体的参考比对结果中发现实际插入的空位非常少，空位率往往是 0.125 甚至更低。例如，1csp_ref1 最长序列长度是 70，参考序列长度是 76，其空位率为(76-70)/70=0.086。如果考虑这样的初始种群，其中个体是不同维的，它们的空位率在 0.125～0.2 之间变化，也许能得到质量更优的比对结果。对于这种随机插入空位且不同维的个体，这里称之为具有随机插入空位变长度的个体，简称 randvar。如果是连续插入空位且不同维的个体，这里称之为具有生物特性变长度的个体，简称 biovar。直接生成带有连续空位的个体作为初始种群个体，将会优化初始种群的个体质量，从而提高比对效率。

2. 加入 MAFFT 种子的初始化

应用在线比对工具 MAFFT，当输入不同的比对参数 gop(gap open penalty)和 gep(gap extension penalty)，可以快速得到不同的质量较好的比对结果。MAFFT 等在线比对工具高度依赖于输入的比对参数，为保持其随机性和多样性，比对时输入 1000 组互不相同的参数，生成 1000 组比对结果，再从中随机选取比对结果作为种子，其个数不应超过初始种群规模。当将这些优质种子以一定比例加入到初始种群中，经过遗传操作和迭代后能得到更好的比对结果。这种加入

优质种子的处理不但增加了初始种群的多样性，同时也从整体上进一步优化了初始种群的质量。

4.2.3　几种初始化方法的构造

初始化时，在序列中随机插入空格得到初始种群的个体。对于每一条序列 S_i 随机选择 x 个位置插入空位“—”，$x = L - l_i$，使得处理后每个序列的长度都为 L。根据插入空位的方式和空位特点，分为以下四种初始化方法。

1. randfix 的构造步骤

(1) 确定插入空位后序列的长度 $L = [len_{max} \times 1.2]$，其中 len_{max} 是每一条序列长度 len_i 的最大值，即最长序列的长度。

(2) 第 i 条序列需插入空位的数目为 $L - len_i$，随机产生不重复的位置，并将空位插入相应位置。

(3) 根据原始序列，按顺序将字符复制到除空位外的相应位置，形成了种群的个体。

(4) 如果个体中有一列全部是空位，则删除该空位列。

生成的个体见图 4.8。

```
MLEG-KVK-WFNSEK-GFG-FIEVEGQDDVFVHF-S-AIQGEGFKTLEE-G-QAVSFEIVE-GNRGPQAANVTKE-A
MS-GKMTGIVKWFNADKGFGFITPDDGSKDVFVHF-SAIQNDGY-KSLDEGQKVS-FTIESGAK-G-PAA-GNVTSL
MATGTVK--WFNAEKGFGFI-AQDGGGPDVFVHYS-A-INATGFRS--LEE-NQVVNFDVTH-GE-GPQAE-NVSPA
K-GTV-KWFSDQKGFGFI--TPDD-GGEDLFVHQ-SGI-RSEGFRSLAEGETV-EFEV-ESGGDGRTKAVDV-TGP-
VL-GTVKWFNVRNGYGFINRND-TKEDV-F-VH-Q-TAIKKNNPRKYLRSVGDGETVEFDVVEGEKGAE-AANVTGP
```

(a) 个体 1

```
-MLEGKVKW-FNSEKGFGFIEVEGQDD-V-FVHFSAIQGEGF-KTLEEGQAVSFEIV-EGN-RGP-QAANVTKEA--
MSGKMTGIVKWFN-ADKGFG-FITPDDGS-KD-VFV-HFSAIQNDGYKSLDEGQKVSFTIESGAKGPAAG-NV-TSL
MATGTVKWFNA-EKGFGFIAQDGG-G--PDVFVHYSAIN-AT-GFRSLE-E-NQV-VNFDVTH-GEG-PQAENVSPA
KGT-VKWF-SDQKGFGF-ITPDDGGEDLFVHQSGIRSEGFRS-LAE-GE-TVEFE-V-ES-GGDGRTKAVDVT-GP-
VLGTVKWFNVRNGYGFINRN--DTKEDV-FVHQT--AIKKNNPRKYLRSVGDGETVEFDVVEGEKGAEAAN-VTGP-
```

(b) 个体 2

图 4.8　1csp_ref1 的两个 randfix 初始化个体

2. randvar 的构造步骤

(1) 随机生成与种群规模个数相同的整数，其范围在 $[len_{max}+1, len_{max}\times 1.2]$ 之间，这些随机整数就是种群每个个体插入空位后序列的长度 L，这样的设置可以使得每个个体的空位率不同。

(2) 对于第 j 个个体的第 i 条序列需插入空位的数目为 $L_j - len_i$，随机产生不重复的位置，并将空位插入相应位置。

(3) 根据原始序列，按顺序将字符复制到除空位外的相应位置，形成了种群的个体。

(4) 删除全空位列。

生成的个体见图 4.9。

```
MLEGKVKW-FNSE-K--GFGFIEVEG-QDDVFVHFSAIQGEGFKTLEEGQAVSFEIVEGNRGPQAANVTKEA
MSGKMTG-IVKWFNADKGFGFITPDDGSKDVFVHFSAIQNDGYKSLDEGQKVSFTIESGAKGPAAGNV-TSL
MATG-TVKWFNAE--KGFGFIAQDGGGPDVFVH-YSAINATGFRSLEENQVV-NFDVT-HGEGPQAENVSPA
K-GTVK-W-FSDQKGFGFITPDDGGEDLF-VHQSGIRSEGFRSLAEGET-VEFEVE-SGGDGRTKAVDVTGP
VLGTVKWFNVRNGYGFINRNDTKEDVFVHQTAIKKNNPRKYLR-SVGDGET-VEFDVVEGEKGAEAANVTGP
```

(a) 个体 1

```
MLEGKVKWFNSEKGFGFIE-VEGQ-DDVFVHFSAIQ--GEG-FKTLEEGQAV-SFEIVEG-N-RGPQAANVT--KEA
-MSGK-MTGIVKWFNADK-GF-GFITPDDGSKDVFVHFS-AIQNDGYKSLDEG-QKVSFTIESGAKGPAA-GNVTSL
-MAT-GTVKWFNAEKGFGFIAQDGGGPDVFVHYSAINATG--FRSL-EENQVVN-FDVTHGEG--PQ-AENV--SPA
-KGTV-KWF-SDQKGFGFI-TPD-DGGEDLF-VHQSGIRSEGFRSLAEGET-VEFEV-ES-GGDG-RTK-AVDVTGP
VLGTVKWFNVRNGY-G-FINRNDTKEDVFVHQ-TAIK-KNNPRKYLRSVGDGETVEFDVVEGEK-G-AEAANVT-GP
```

(b) 个体 2

图 4.9　1csp_ref1 的两个 randvar 初始化个体

3. biofix 的构造步骤

(1) 确定插入空位后序列的长度 $L=[len_{max}\times 1.2]$，其中 len_{max} 是每一条序列长度 len_i 的最大值，即最长序列的长度。

(2) 假设插入的每一段空位长度至少大于 2，根据需插入空位的长度 $L-len_i$ 确定需要插入多少段连续空位，随机产生不重复的位置，并将连续空位插入相应位置，如果空位长度不能整除连续空位长度，

则余数为剩余空位，在非空位处随机插入剩余空位。

(3) 根据原始序列，按顺序将字符复制到除空位外的相应位置。

(4) 删除全空位列。

生成的个体见图 4.10。

```
--MLEGKVKWFNSEKGFGFIEVEGQD--DVFVHF--SAIQGEGFKTLEEGQAVSFEI----VEGNRGPQAANVTKEA
MSGKM-TGIVK--WF--NADKGFGFITPDDGSKDVFV--HFSAIQNDGYKSLDEGQKVSFTIESGAKGPAAGNVTSL
MAT--GTVKWFNA--EKGFGFIAQD--GGGPDVFVHYSAI-NATGFRSLEENQVVNF--D--VTHGEGPQAENVSPA
KGTVK--WFSDQKGFGFITPDD--GGEDLFVHQSGIR--SEGFRSL--AEGETVEFEVESG--GDGRTKAVD-VTGP
VLGTVKWFNVRNGYG--FINRNDTKE--DVFVHQTAIK--KNNPRKYLR-SVGDGETVEFDVVEGEKGAEAANVTGP
```

(a) 个体 1

```
MLEGKVKWFNSEK--GFGFIEVEGQDDV--FVH--FSAIQG--EGFKTLEEGQAVSFEIVEGNRGPQAAN--VTKEA
MSGKMTGIV--KWFNAD-KGFGF--ITPDDG--SKDVFVHFSAIQNDGYKSLDEGQKVSFTIESGAKGPAAGNVTSL
MATGTVKWFNAEK--GFG--FI--AQDGG--GPDVFVH-YSAINATGFRSLEE--NQVVNFDVTHGEGPQAENVSPA
----KGTVKWFSDQKG--FGF--ITPD-D--GGEDLFVHQSGIRSEGFRSLAEGETVEFEVESGGDGRTKAVDVTGP
--VLGTVKWFNVRNGYGFIN-RNDTKEDVFVHQTAIKKNNPR--KYLRSVGDGETV--EFDVVEGEKGAEAANVTGP
```

(b) 个体 2

图 4.10　1csp_ref1 的两个 biofix 初始化个体

4. biovar 的构造步骤

(1) 随机生成与种群规模个数相同的整数，其大小范围在 $[len_{max}+1, len_{max}\times 1.2]$ 之间，这些随机整数就是种群每个个体插入空位后序列的长度 L。

(2) 假设插入的每一段空位长度至少大于 2，根据需插入空位的长度 $L-len_i$ 确定需要插入多少段连续空位，随机产生不重复的位置，并将连续空位插入相应位置，如果空位长度不能整除连续空位长度，则余数为剩余空位，在非空位处随机插入剩余空位。

(3) 根据原始序列，按顺序将字符复制到除空位外的相应位置。

(4) 删除全空位列。

生成的个体见图 4.11。

通过以上个体图可以看出，当采用固定长度的初始化时，插入的空位长度是统一的，所以生成的个体维数相同，如图 4.8 和图 4.10

所示。而当采用变长度的初始化时，插入的空位长度是随机的，所以不同个体的长度是不同的，如图 4.9 和图 4.11 所示，这样的处理更能增加初始种群个体多样性。

```
MLEGKVKWFNSEK--GFGFIEVEGQDDVFVHFSAIQGEGFKTLEEG--Q-AVSFEIVEGNRG--PQAANVTKEA
MSGKMTGIVKW--FNADKGFGFITPDDGSKDVFVHFSAIQNDGYKSLDEGQKVSFTIESGAKGPAAGNV--TSL
M--ATGTVKWFNAEKGFGFIAQDGGGPDV--FVHYSAINA--TGFRSLEENQVV--NFDVTHGEGPQAENVSPA
--KGTVKWFSDQKGFGFITPDDGG--EDLFVHQSGIRSEGFRSLAEGE--TV--EFEVESGGDGRTKAVDVTGP
VLGT--VKWFNVRNGYGFINRNDTKEDVFVHQTAIKKNNPRKYLRSVGDGETVEFDVVEGEK--GAEAANVTGP
```

(a) 个体 1

```
MLEGKVKWFN--SEKGFGFIEVEGQ--DDVFVHFSAIQ--G---EGFKTLEEGQAVSFEIVE--GNRGPQAANVTKEA
MSGKMTGIVKWFNADKG----FGFITPDDGSKDVFV--HFSAIQNDGYKSLD--EGQKVSFTIESGAKGPAAGNVTSL
--MATGTVKWFNAEKGFGFIAQDG--G--GPDVFVHYSAIN--ATGFRSLEENQVVNFDVTHG--EGPQAENVSP--A
KGTVKWFS--DQKGFGFIT--PDDGGEDLFVH--Q--SGI--RSEGFRSLAEGETVEFEVESGGD--GRTKAVDVTGP
VLGTVKW--FNVRNGYGFINRNDTKEDVF--VHQTAIKKNNPRKYLRSVGDGETV----EFDVVEGEKGAEAANVTGP
```

(b) 个体 2

图 4.11　1csp_ref1 的两个 biovar 初始化个体

4.2.4　加入 MAFFT 种子的初始化

应用在线比对工具 MAFFT，输入 1000 组互不相同的比对参数 gop(gap open penalty)和 gep(gap extension penalty)，可以得到 1000 组不同的比对结果。假设初始种群规模是 50 个个体，则从 1000 组比对结果中随机选取 50 个 MAFFT 比对结果作为优秀种子，如果将这 50 个优秀种子全部作为初始种群的个体，则该种群简称为 50MAFFT 种子。

如果从这 50 个优秀种子中再随机选出 20 个，并随机替换上面的随机初始种群中的 20 个个体，则该种群简称为 20MAFFT 种子。如果原种群中不插入优秀种子，即原随机种群，简称为 0MAFFT 种子。

4.2.5　实验算例与结果

1. 实验参数

应用改进遗传算法模拟多序列比对的实验参数汇总见表 4.4。

表 4.4　实验参数

Parameter	Dataset10	Fitness function	Sum-of-pairs
Population size	50	Score matrix	BLOSUM45
Generations	3000	Crossover operator	One point
Selection strategy	Roulette wheel	Crossover probability	0.6
Elitist rate	0.1	Mutation operator	One bit
Programming language	MATLAB	Mutation probability	0.2
Coding	Two-dimensional		

2. 实验结果与分析

从 BAliBASE2.0 数据库中随机选了 10 组多序列，以 SPS 作为评估比对质量的标准。由于遗传算法具有随机性，对相同的实验用例可能得到不同的比对结果，所以这里对每个实验用例用同样的方法做 10 次，计算其 SPS 值，再取其最大值为最终的结果。

四种初始化的序列比对 SPS 值如表 4.5 所示，最优解分布情况及曲线如图 4.12 所示。

表 4.5　四种初始化的序列比对 SPS 值

(黑体数字表示该组序列的 SPS 最优解)

名称	SPS			
	randfix	randvar	biofix	biovar
1aab_ref1	0.3543	**0.4397**	**0.4347**	0.3794
1aboA_ref1	0.2253	**0.2619**	0.185	0.2491
1aho_ref1	0.4889	0.5231	0.5573	**0.6291**
1csp_ref1	0.5315	0.5945	0.7112	**0.768**
1dox_ref1	0.388	0.4324	**0.4479**	0.4093
1r69_ref1	0.1835	0.1782	0.1915	**0.3378**
1ubi_ref1	0.2357	0.1929	0.2333	**0.2548**
1ycc_ref1	0.2653	**0.3752**	0.3264	**0.3752**
2trx_ref1	0.1911	**0.2857**	0.1288	0.173
451c_ref1	0.3493	0.3534	**0.4082**	0.337
Max_number	0	4	3	5

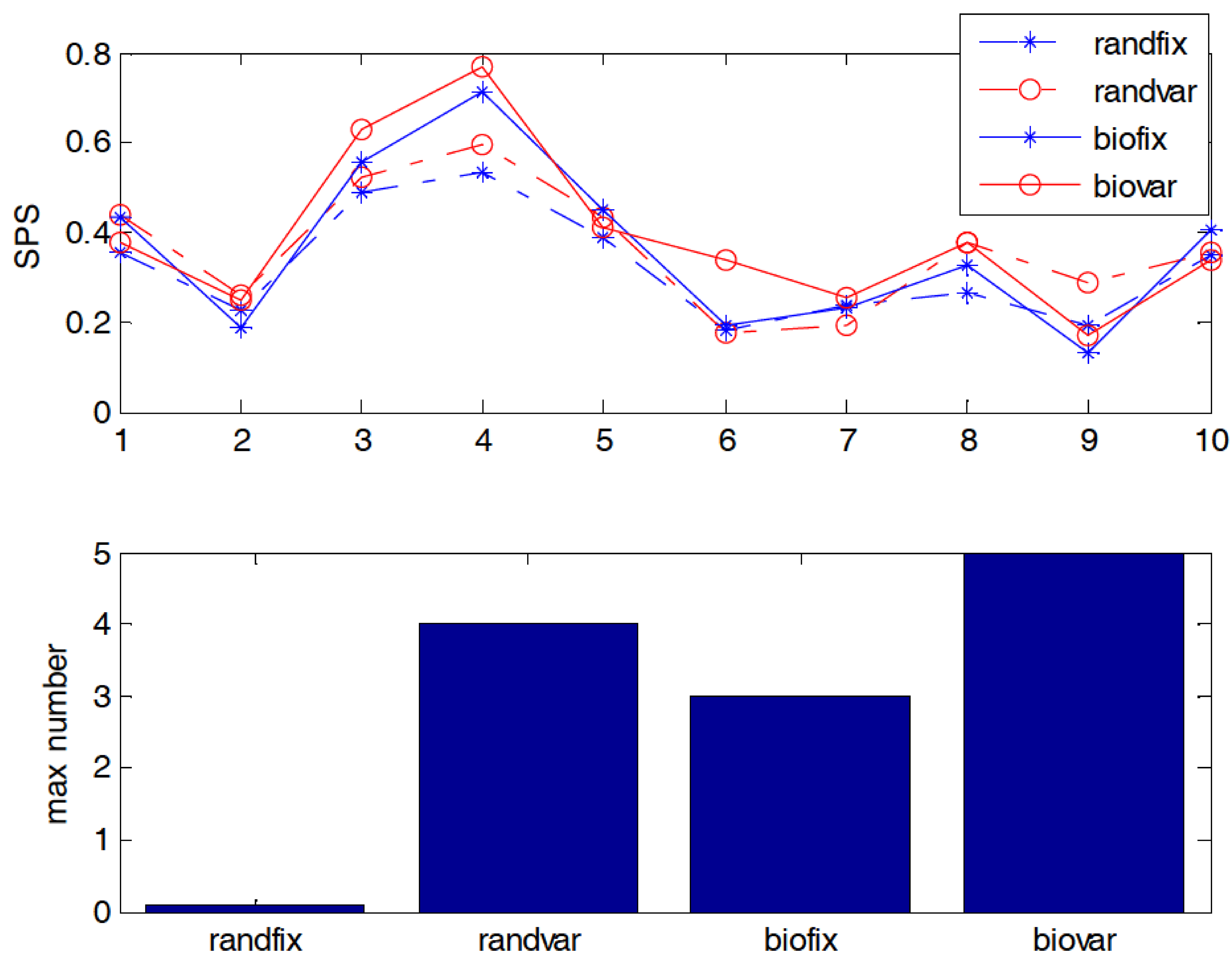


图 4.12　四种初始化的序列比对 SPS 值

从表 4.5 和图 4.12 可以看出：

(1) 种群中变长度个体比固定长度个体的计算效果好。

(2) 具有生物特性的初始化比随机初始化的计算效果好。

1csp_ref1 手动比对好的参考序列和应用本文算法比对的结果见图 4.13 和图 4.14，其中*是全对齐列。

```
---MLEGKVKWFNSEKGFGFIEV-EGQDDVFVHFSAIQG----EGFKTLEEGQAVSFEIVEGNRG-PQAANVTKEA
MSGKMTGIVKWFNADKGFGFITPDDGSKDVFVHFSAIQN----DGYKSLDEGQKVSFTIESGAKG-PAAGNVTSL-
---MATGTVKWFNAEKGFGFIAQDGGGPDVFVHYSAINA----TGFRSLEENQVVNFDVTHG-EG-PQAENVSPA-
-----KGTVKWFSDQKGFGFITPDDGGEDLFVHQSGIRS----EGFRSLAEGETVEFEVESGGDGRTKAVDVTGP-
----VLGTVKWFNVRNGYGFINRNDTKEDVFVHQTAIKKNNPRKYLRSVGDGETVEFDVVEGEKG-AEAANVTGP-
      * ****    * ***       * ***   *                 * *    *  *   *  *
```

图 4.13　1csp_ref1 的参考序列

```
MLEG-K--VKWFNSEKGFGFIEVE-GQDDVFVHFSAIQGE-GFKTLEE-GQA--VSFEIVEGNRGPQ-AANVTKEA
MSGKMTGIVKWFNADKGFGFITPDDGSKDVFVHFSAIQ-NDG--Y-KSLDEGQKVSFTIESGAKG-P-AAGNVTSL
MATG---TVKWFNAEKGFGFIAQDGGGPDVFVHYSAINAT-G-F--RSLEENQVVNFDV-T--HGEGPQAENVSPA
--KG---TVKWFSDQKGFGFITPDDGGEDLFVHQSGIRSE-G--F-RSLAEGETVEFEVESGGDGRTKAVD-VTGP
VL-G---TVKWFNVRNGYGFINRNDTKEDVFVHQTAIKKNNPRKYLRSVGDGETVEFDVVEGEKG-AEAA-NVTGP
        ****    * ***       * ***   *                 * *       *
```

图 4.14　1csp_ref1 比对结果

结论：在序列中插入不同数目的连续空位，形成包含不同维个体的初始种群，不但具有生物意义，也增加了初始种群的多样性；通过实验算例的结果比较，这种插入连续空位的不同维个体最适合作为多序列比对的初始种群。

从 BAliBASE2.0 数据库中随机选了 10 组多序列，以 SPS 作为评估比对质量的标准。由于遗传算法具有随机性，对相同的实验用例可能得到不同的比对结果，所以这里对每个实验用例用同样的方法做 10 次，计算其 SPS 值，再取其平均值和最大值作为最终的结果，见表 4.6、图 4.15 和表 4.7、图 4.16。

这是种群规模 50，进化代数 3000 次的结果。

表 4.6　三种加入 MAFFT 优秀种子的序列比对 SPS 平均值

数据库名称	长度(bp)	SPS(horizontal cross)			SPS (vertical cross)		
		0M	20M	50M	0M	20M	50M
451c_ref1	87	0.3005	0.3668	0.3851	0.3661	0.3616	0.3801
1aab_ref1	79	0.3436	0.4265	0.4246	0.2682	0.4246	0.4139
1aho_ref1	65	0.5549	0.5613	0.5212	0.4714	0.5632	0.5184
1pfc_ref1	116	0.4742	0.5076	0.4895	0.4316	0.5065	0.4956
2cba_ref1	259	0.1781	0.5325	0.5325	0.1549	0.5325	0.492
2pia_ref1	287	0.0785	0.2687	0.2683	0.0486	0.1939	0.2337
5ptp_ref1	245	0.4162	0.9066	0.9165	0.3897	0.9066	0.9165
kinase_ref1	276	0.1922	0.4755	0.4622	0.1959	0.4361	0.4823
时间(s)		7800			10000		

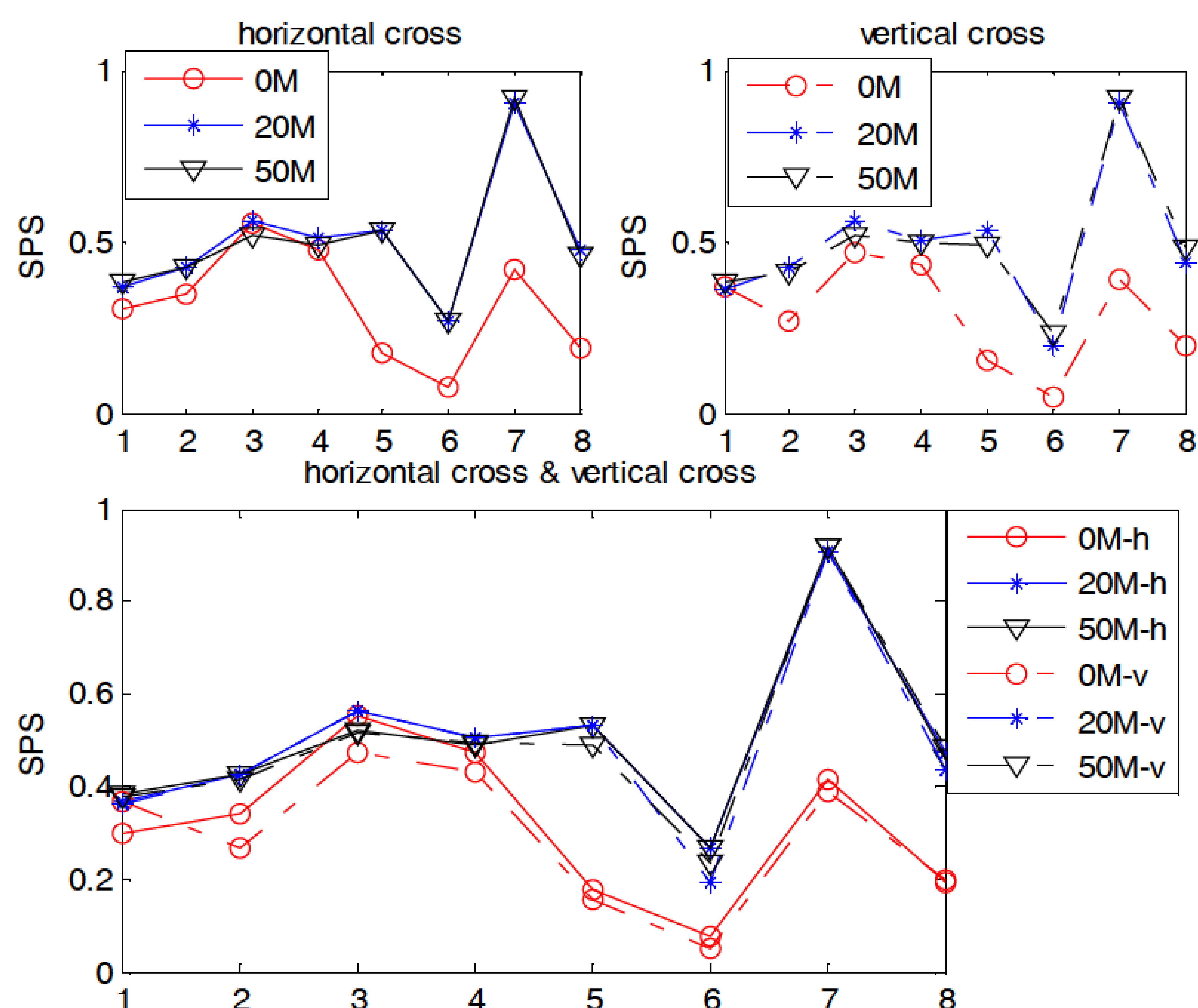


图 4.15　三种加入 MAFFT 优秀种子的序列比对 SPS 平均值

表 4.7　三种加入 MAFFT 优秀种子的序列比对 SPS 最大值

(**黑体数字**表示该组序列的 SPS 最优解)

数据库名称	长度(bp)	SPS(horizontal cross)			SPS (vertical cross)		
		0M	20M	50M	0M	20M	50M
451c_ref1	87	0.4096	0.3740	0.4041	**0.4329**	0.3753	0.4082
1aab_ref1	79	**0.4623**	0.4397	0.4246	0.3568	0.4246	0.4246
1aho_ref1	65	**0.6957**	0.6462	0.6479	0.5590	0.6205	0.5726
1pfc_ref1	116	0.5559	**0.5745**	0.4981	0.4926	0.5158	0.5084
2cba_ref1	259	0.2223	**0.5325**	**0.5325**	0.2015	**0.5325**	0.5216
2pia_ref1	287	0.1236	**0.2875**	**0.2875**	0.0614	0.2487	0.2436
5ptp_ref1	245	0.4495	0.9066	**0.9165**	0.4341	0.9066	**0.9165**
kinase_ref1	276	0.2386	0.4755	0.4892	0.2153	0.4755	**0.5016**
时间(s)		7800			10000		

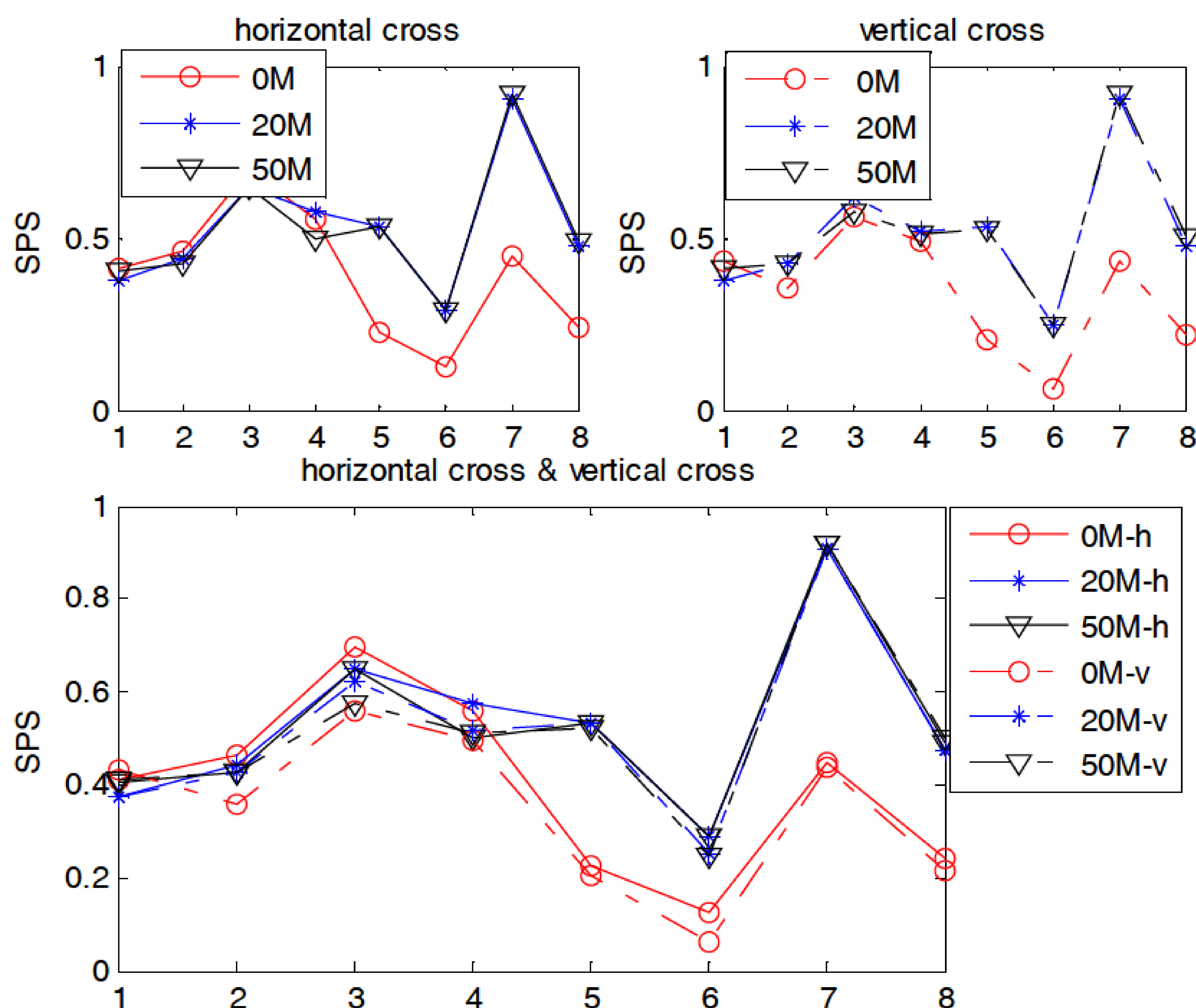


图 4.16　三种加入 MAFFT 优秀种子的序列比对 SPS 最大值

从以上数据可以看出：

(1) 横向交叉比纵向交叉的计算效果好。

(2) 当序列长度<120bp 时，0 种子/20 种子/50 种子的 SPS 值相差不大，最好选择 0M。

(3) 当序列长度>120bp 时，0 种子的分值与 20 种子/50 种子的 SPS 值相差很大。优先选择 20M 和 50M，但是 50M 的计算费用较高，且容易陷入局部最优解，故最好选择 20M。

(4) 对于遗传算法这种随机算法来说，虽然平均值数据可以反映出算法的性能，但是最后的比对结果仍以最大值为准。从两个图可以看出平均值与最大值的趋势走向差不多，因此，以后的遗传算法数据均只计算最大值。

4.2.6　结论

遗传算法可以有效解决生物多序列比对问题，但是遗传算法高

度依赖于初始种群，好的初始种群可以得到好的结果。为提高计算效率，提高比对质量，本节从遗传算法最关键的组成部分入手，通过优化初始种群的质量，达到改进算法的目的。

在序列中插入不同数目的连续空位，形成包含不同维个体的初始种群，这样生成的个体不但具有生物意义，也增加了初始种群的多样性，优化了初始种群的个体质量；在这个初始种群的基础上，加入适当比例的优秀 MAFFT 种子，不会陷入局部最优解，优化了初始种群的整体质量。因为多序列比对通常的比对对象是长度>120bp 的中长序列，这种插入连续空位的不同维个体最适合作为多序列比对的原始初始种群。对于中长序列比对问题，在原始初始种群中加入 2∶5 的 MAFFT 优质种子，可以达到最好的计算效率。通过实验验证，这两个做法的组合优化了初始种群的质量，提高了多序列比对的计算效率，从而达到改进的目的。

4.3　改进遗传算法之交叉算子优化

4.3.1　引言

交叉算子是构造遗传算法的一个关键部分。Notredame 共设置了交叉、加空位、移动空位等 22 个遗传算子；Naznin 又提出单点交叉与多点交叉；Fan 提出智能算子在遗传算法中的应用。有的遗传算子构造较为复杂，但是 Thomsen 验证了简单或复杂的交叉算子对于比对结果没有明显区别。Goondro 认为无须设计过分复杂的算子，遗传算法就可以满足基本的比对需求，因此单点交叉成为目前最常用的交叉算子。按照交叉方式，单点交叉属于整体纵向交叉，由于序列比对要求交叉后的残基顺序要保持不变，所以这种纵向交叉的编程难度较大。本节又提出三种构造简单的交叉方式，试图降低编程难度。

在交叉操作中，还有一个后处理的过程，即如何确定交叉后的子代，一般的做法是交叉后的结果就是子代。本节对于这个后处理进行优化处理，提出了改进方法 cross4to2。

本节应用基本遗传算法对多序列进行比对模拟，并对遗传算法中最基本的交叉操作与其后处理进行优化，通过实验比较，结果表明多行横向交叉的计算效果最好，后处理方式 cross4to2 能有效缩短计算时间，二者相结合能明显提高遗传算法的计算效率。

4.3.2　交叉算子设计

1. 单点纵向交叉法

在种群中随机配对，根据交叉概率选定某对进行交叉，在父代个体 1 中随机设定一个交叉点，在父代个体 2 中找到相应的交叉点，实行交叉时，该点前或后的两个个体的部分结构进行交换，并生成两个新个体，如果维数不同，则在末端补齐空位。示意图见图 4.17。

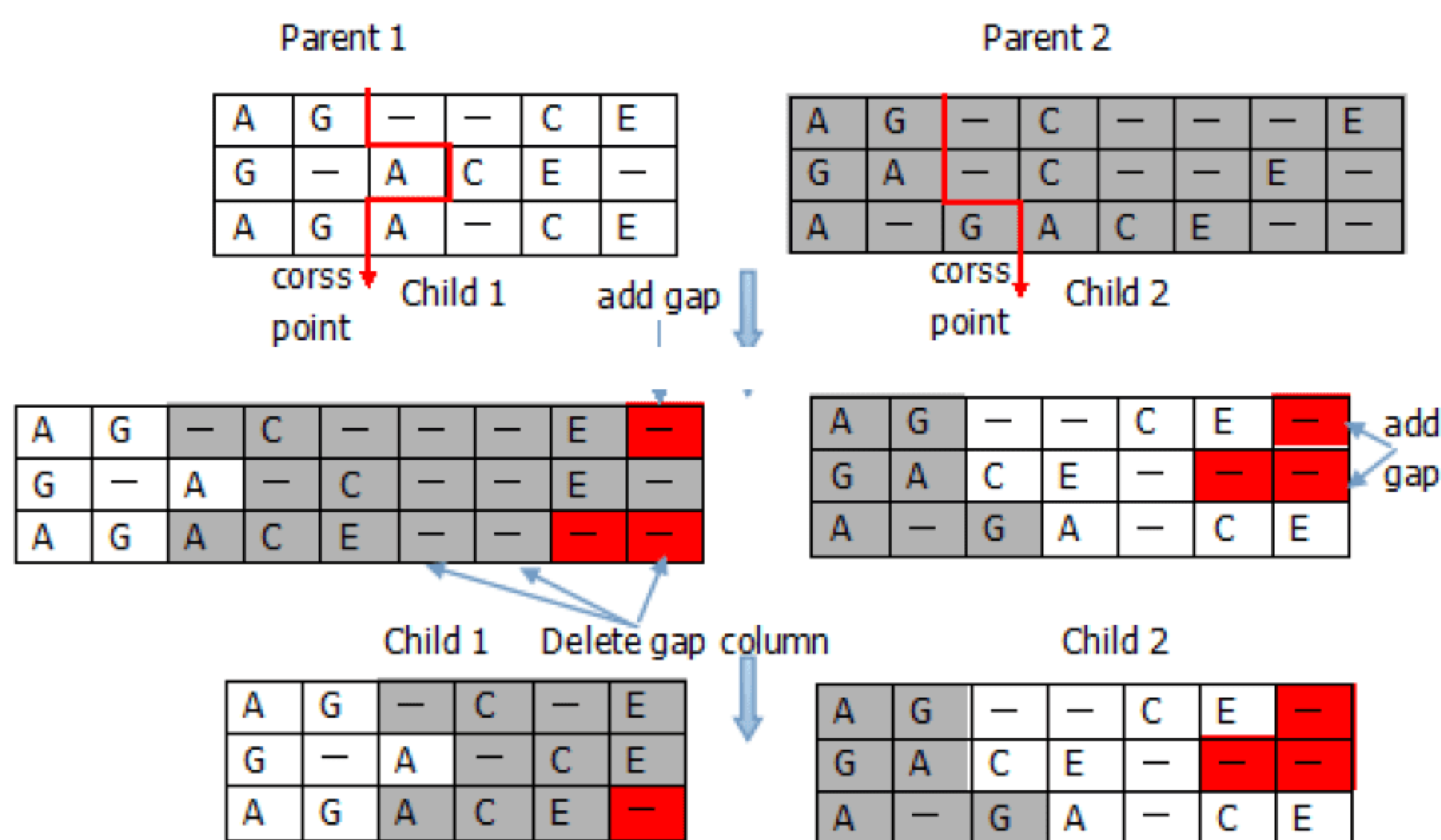


图 4.17　单点纵向交叉

2. 单点横向交叉法

两个父体之间仅单行参与交叉。产生一个不超过序列行数的随

机数，在该行实行交叉，如果两个父体是不同维的，则在短序列后面补齐空位，然后删除全空位列。示意图见图 4.18。

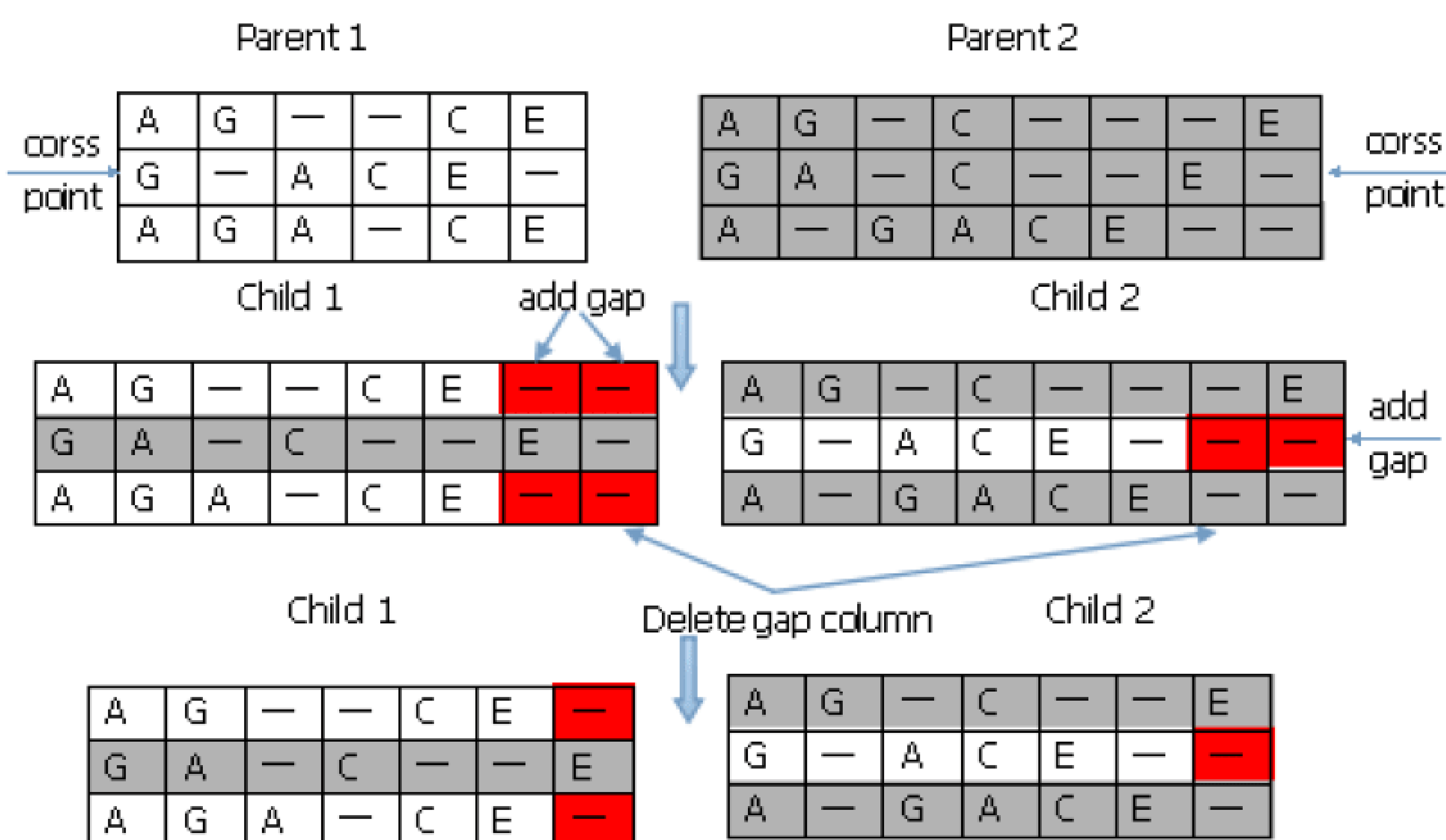


图 4.18　单点横向交叉

3. 多行横向交叉法

两个父体之间有多行参与交叉。产生一个不超过序列行数的随机数，在该行及以下所有行实行交叉。不同维交叉是在短序列后面补齐空位，然后删除全空位列。示意图见图 4.19。

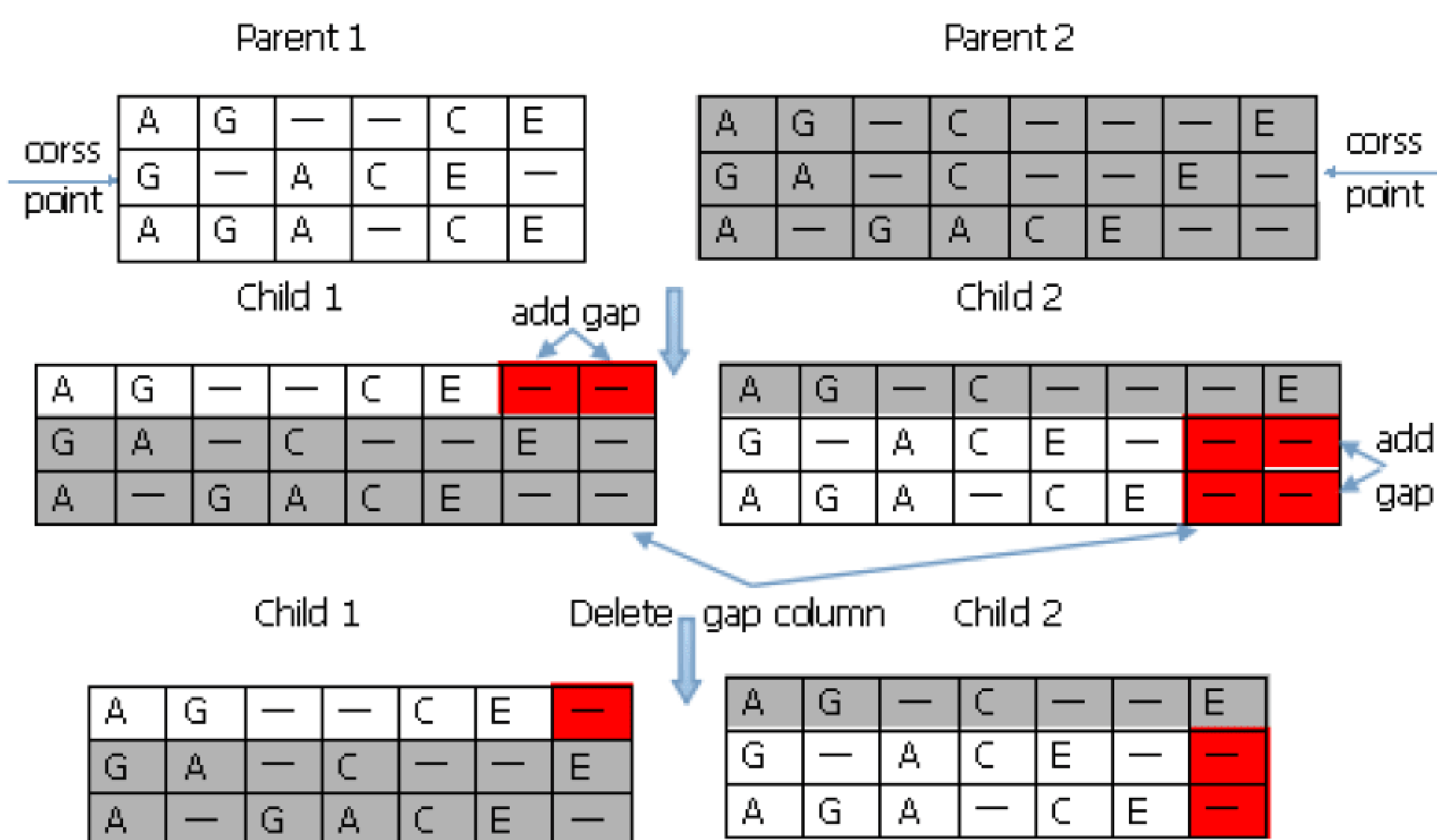


图 4.19　多行横向交叉

4. 单点横向交叉变异法

两个父体之间仅单行参与交叉。产生一个不超过序列行数的随机数，在该行实行交叉，如果两个父体是不同维的，则在短序列后面补齐空位，在该行中随机选择一个空位和一个字符进行变异，将空位移至字符位置，将选定字符位与空位之间的这段向后(前)顺延一位，然后删除全空位列。示意图见图 4.20。

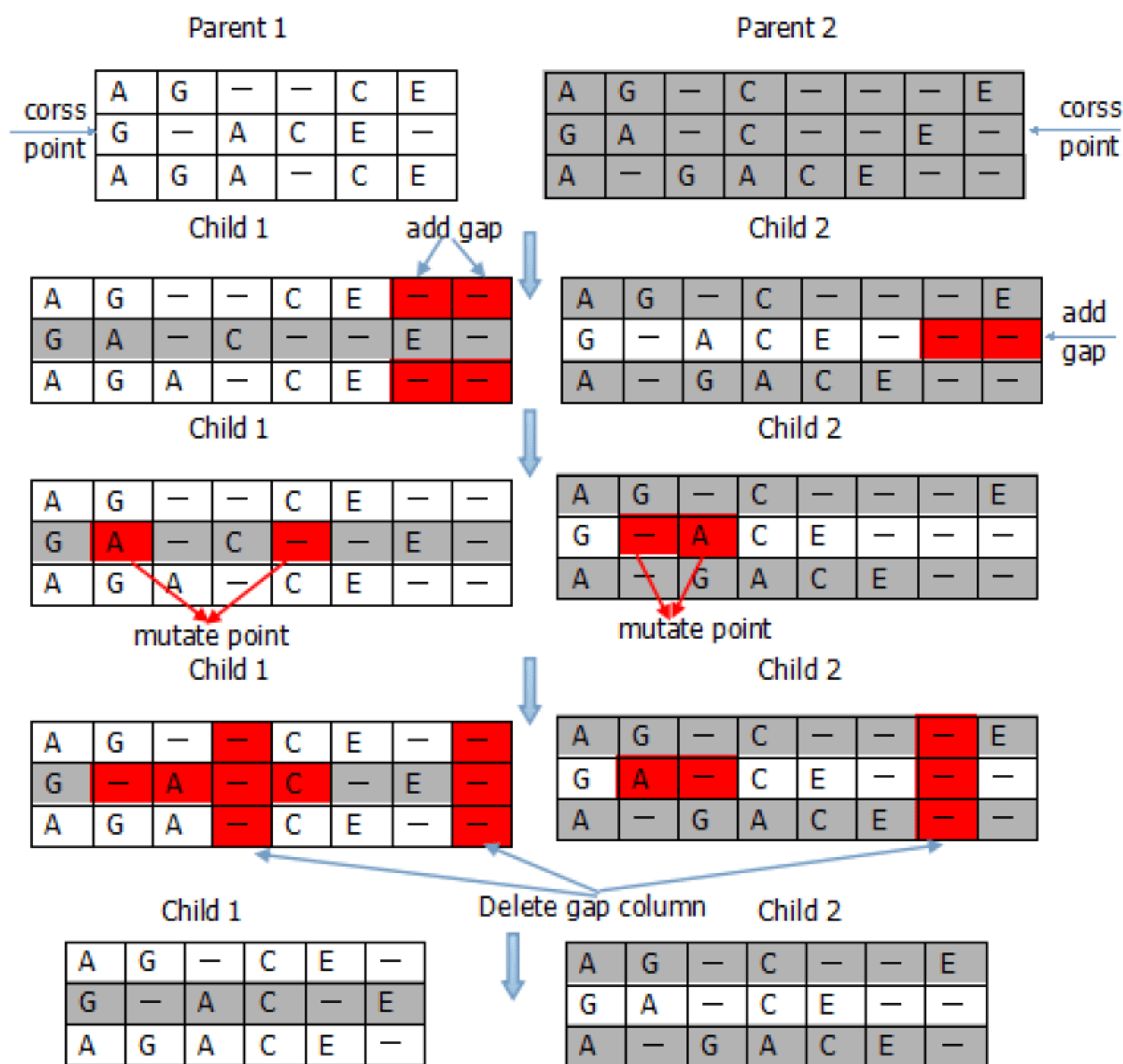


图 4.20　单点横向交叉变异

5. 交叉操作后处理

因为交叉算子需要 2 个父本，经交叉产生 2 个新个体，为保持种群规模不变，需从中选择 2 个个体作为新一代。这种选择新一代的过程称为交叉操作后处理。

常见的后处理方式有以下几种：

(1) cross1to1。子代序列 1、2 替换父代序列 1、2，直接作为新一代，见图 4.21。

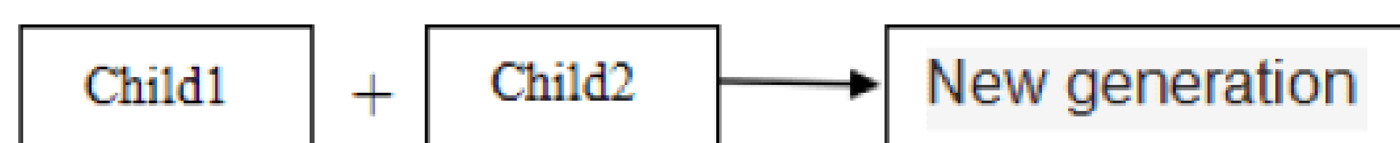


图 4.21　cross1to1 过程图

(2) cross2to1。比较父代序列 1 和子代序列 1 的适应度值，取高的值作为下一代，即保优原则，见图 4.22。

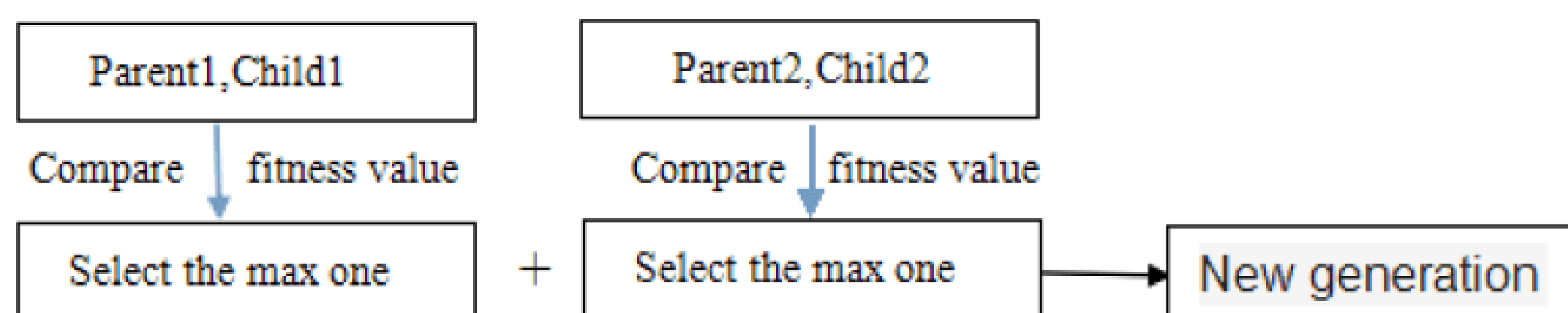


图 4.22　cross2to1 过程图

(3) cross4to2。比较父代序列 1、2 和子代序列 1、2，从中选择适应度值最高的 2 个个体作为下一代，这个方式综合了保优原则和选择精英保留原则，算法构造较简单，见图 4.23。

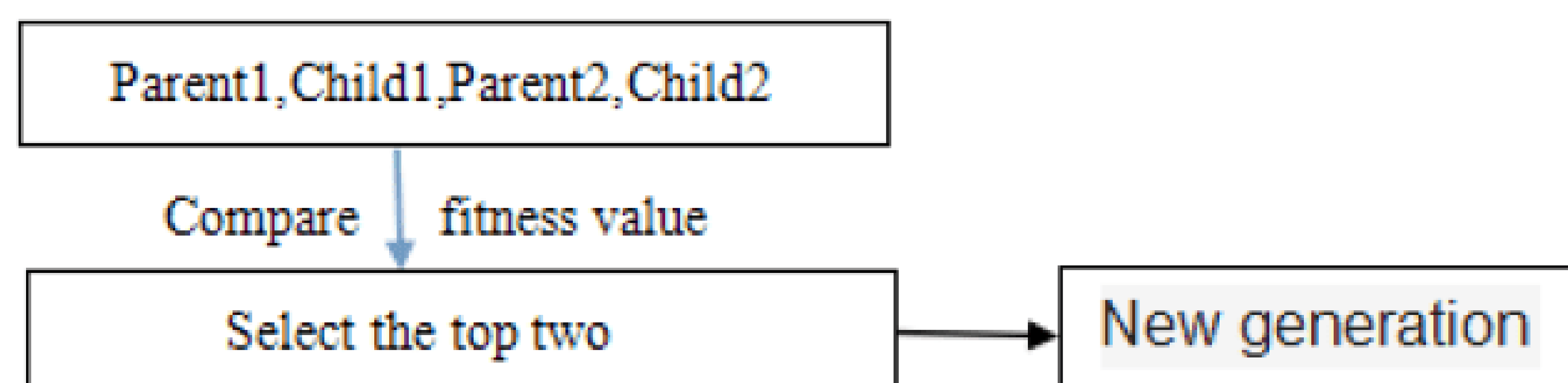


图 4.23　cross4to2 过程图

4.3.3　实验算例与结果

1. 实验参数

改进交叉算子的遗传算法模拟多序列比对的实验参数汇总见表 4.8。

表 4.8　实验参数

Parameter	Dataset8	Fitness function	Sum-of-pairs
Population size	50	Score matrix	BLOSUM45
Generations	3000	Crossover operator	One point
Selection strategy	Roulette wheel	Crossover probability	0.6
Elitist rate	0.1	Mutation operator	One bit
Programming language	MATLAB	Mutation probability	0.2
Coding	Two-dimensional		

2. 实验结果与分析

从 BAliBASE2.0 数据库中随机选了 8 组多序列，以 SPS 作为评估比对质量的标准。由于遗传算法具有随机性，对相同的实验用例可能得到不同的比对结果，所以本节对每个实验用例用同样的方法做 10 次，计算其 SPS 值，再取其最大值作为最终的结果。表 4.9 与图 4.24 中：单点纵向交叉简写为 ver，单点横向交叉简写为 hor，多行横向交叉简写为 mhor，单点横向交叉变异简写为 c+m。

表 4.9　BAliBASE2.01 8 组序列的 SPS 值

(**黑体数字**是该组序列的 SPS 最大值)

名称	SPS(horizontal cross)			
	ver	hor	mhor	c+m
451c_ref1	0.2688	0.2437	**0.3191**	0.3015
1aab_ref1	0.2363	**0.3168**	0.2985	0.2198
1aho_ref1	0.4974	0.4735	**0.5641**	0.5419
1pfc_ref1	0.4531	0.5515	**0.616**	0.3456
2cba_ref1	0.2394	0.2703	**0.4247**	0.2954
2pia_ref1	0.2952	0.3191	**0.3723**	0.1941
5ptp_ref1	**0.2571**	0.1976	0.2262	0.181
kinase_ref1	0.3194	0.2443	**0.3229**	0.2408
Time(s)	4649	3727	3745	3217

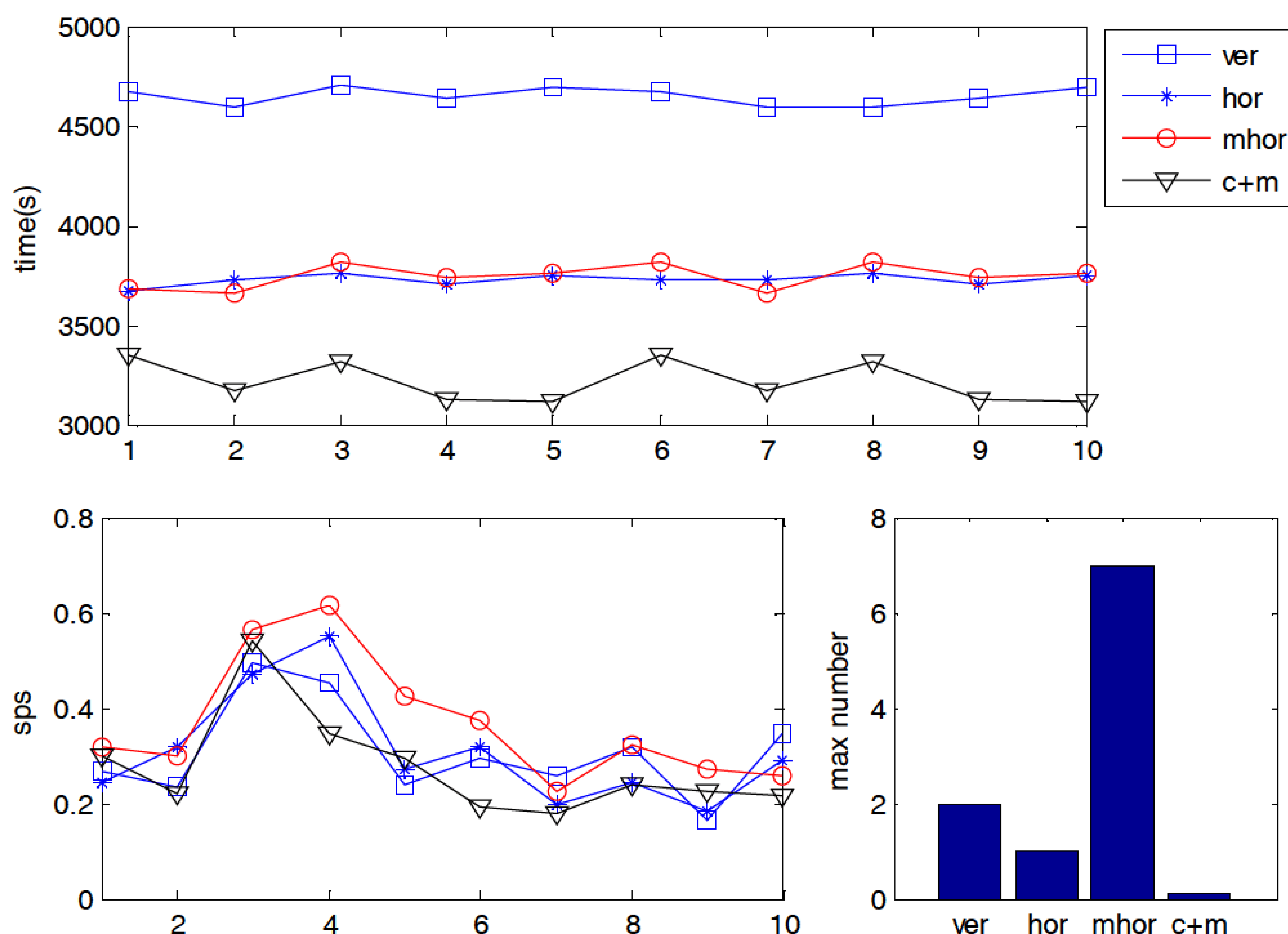


图 4.24　四种交叉算子的 SPS 比较图

通过表 4.9 和图 4.24 可知，纵向交叉的计算时间最多，c+m 的计算时间最少，两种横向交叉的时间差不多。SPS 值最好的是多行横向交叉，最差的是 c+m。因为 c+m 方式不再一次变异，虽然节省了时间，但也影响了比对结果。总之，根据计算耗时与计算结果，得知多行横向交叉的计算效率最好。

通过图 4.25 可知，在相同的迭代次数下，未经过保优处理的 cross1to1 的计算效率最低，保优处理的 cross2to1 适应度值较高，但是容易局部收敛。保优与选择综合的 cross4to2 的分值远高于另两种后处理方式，说明改进算法有效。如果将多行横向交叉与 cross4to2 结合起来，将大大提高计算效率。

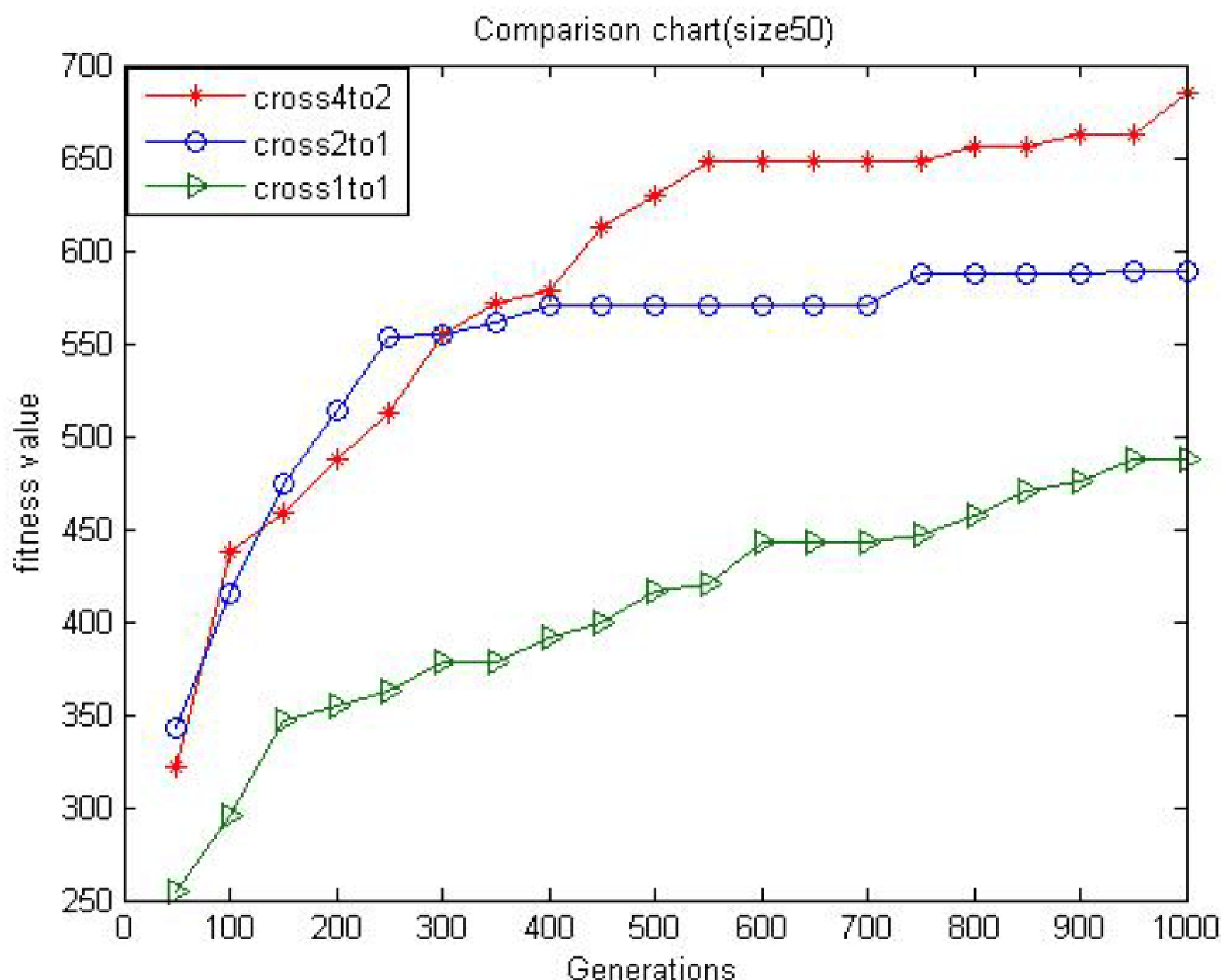


图 4.25 三种交叉后处理的适应度值比较图

4.3.4 结论

遗传算法是近几年较热门的多序列比对方法，本节针对遗传算法最基本的交叉算子，设计了保优和选择混合的交叉操作后处理方法 cross4to2，该方法不但服从保优原则，而且又再一次经过选择操作的精英保留过程，使得最优秀的个体进入下一代，这种处理将算法的整体搜索能力和局部搜索能力大大提高。通过与经典 CLUSTAL 算法的比较，验证了该算法的有效性。从理论分析和实例可见，笔者提出的混合后处理 cross4to2 在 MSA 问题求解上得到满意的结果，同时，该方法应能推广到遗传算法在别的领域中的计算。

遗传算法有两个基本要素：种群规模和进化代数。种群规模决定种群多样性和算法收敛性，规模太小则不能保证多样性导致“早熟”现象，规模太大则耗费过多的计算时间。进化代数越高，计算精度越好，同样和计算时间成正比。本节讨论了种群规模与进化代

数的关系，在种群中有 50 个个体时，能保证算法正常收敛，提高计算分值和精度只需适当增加进化代数即可，这种限制种群规模增加进化代数的处理使得算法的性能进一步提高。

4.4　本章小结

本章应用基本遗传算法及其改进的遗传算法进行多序列比对。基本遗传算法(GA)是通过对进化过程中的种群反复进行选择、交叉、变异操作来模拟自然界中种群的演变过程，直到满足一定性能要求才结束计算，它本身的结构决定了它可以用在多序列比对上。遗传算法可以有效解决生物多序列比对问题，但是遗传算法高度依赖于初始种群，好的初始种群可以得到好的结果。为提高计算效率，提高比对质量，本章从遗传算法最关键的组成部分入手，通过优化初始种群的质量，达到改进算法的目的。另外，本章又针对遗传算法最基本的交叉算子，设计了保优和选择混合的交叉操作后处理方法 cross4to2，该方法不但服从保优原则，而且又再一次经过选择操作的精英保留过程，使得最优秀的个体进入下一代，这种处理将算法的整体搜索能力和局部搜索能力大大提高。通过与经典 CLUSTAL 算法的比较，验证了该算法的有效性。从理论分析和实例可见，笔者提出的混合后处理 cross4to2 在 MSA 问题求解上得到满意的结果。

为适应当前生物信息的海量性特征以及生物序列爆炸性增长的趋势，研究和设计速度更快、精度更高、效率更好的改进遗传算法极为重要，这也是笔者下一步的工作。

参考文献

[1] 张春霆. 生物信息学的现状与展望[J]. 世界科技研究与发

展, 2000, 22(6): 17-20.

[2] Chuong B, Do, Katoh K. Protein multiple sequence alignment[M]. Functional Proteomics. Humana Press, 2008: 379-413.

[3] 邹权，郭茂祖，韩英鹏. 多序列比对算法的研究进展[J]. 生物信息学，2010 (4)：311-315.

[4] Notredame C. Recent progress in multiple sequence alignment: a survey[J]. Pharmacogenomics, 2002, 3(1): 131-144.

[5] 张鹏帅，霍红卫. 多序列比对问题的粒子群优化算法求解[J]. 计算机工程与应用，2005，41(18)：84-87.

[6] Gusfield D. Algorithms on strings, trees and sequences: computer science and computational biology[M]. Cambridge university press, 1997.

[7] Thompson J D, Higgins D G, Gibson T J. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice[J]. Nucleic acids research, 1994, 22(22): 4673-4680.

[8] 程灏. 混合遗传算法在图着色及 MSA 问题中的应用[D]. 西安电子科技大学，2006.

[9] 林秋利. 基于进化算法和量子计算的多序列比对方法研究[D]. 西安电子科技大学，2006.

[10] Dan D B, Kececioglu J. Parameter advising for multiple sequence alignment[J]. Bmc Bioinformatics, 2015, 16(1):516 - 518.

[11] Gondro C, Kinghorn BP. A simple genetic algorithm for multiple sequence alignment[J]. Genetics and Molecular Research, 2007, 6(4): 964-982.

[12] Wang L, Jiang T. On the complexity of multiple sequence alignment[J]. Journal of Computational Biology, 1994(1):337-348.

[13] 张敏. 生物信息学中多序列比对等算法的研究[D]. 大连理工大学，2005.

[14] Notredame C, Higgins DG, Heringa J. T-COFFEE: a novel

method for fast andaccurate multiple sequence alignments[J]. J.Mol. Evol., 2000, 302(1): 205-217.

[15] 韦树烽，刘羽，蒋财运. 基于GPU的遗传退火多序列比对并行研究[J]. 计算机工程与设计，2014，35(4)：1247-1252.

[16] Kaya M, Sarhan A, Alhajj R. Multiple sequence alignment with affine gap by using multi-objective genetic algorithm[J]. Computer methods and programs in biomedicine, 2014, 114(1): 38-49.

[17] Kwan M, Xiao N, Ding G. Assessing Activity Pattern Similarity with Multidimensional Sequence Alignment Based on a Multiobjective Optimization Evolutionary Algorithm[J]. Geographical Analysis, 2014, 46(3): 297-320.

[18] Notredame C, Higgins D G. SAGA: sequence alignment by genetic algorithm[J]. Nucleic acids research, 1996, 24(8): 1515-1524.

[19] 葛宏伟，梁艳春. 基于隐马尔可夫模型和免疫粒子群优化的多序列比对算法[J]. 计算机研究与发展，2006，43(8)：1330-1336.

[20] Chen W, Liao B, Zhu W, et al. An ant colony pairwise alignment based on the dot plots[J]. Journal of Computational Chemistry, 2009, 30(1): 93-97.

[21] Silva F J M, Sánchez Pérez J M, Gómez Pulido J A, et al. AlineaGA——a genetic algorithm with local search optimization for multiple sequence alignment[J]. Applied Intelligence, 2010, 32(2): 164-172.

[22] Silva F J M, Sánchez Pérez J M, Antonio J, et al. An evolutionary approach for performing multiple sequence alignment[C]. WCCI 2010 IEEE World Congress on Computational Intelligence, Barcelona, Spain, July. 2010: 18-23.

[23] Rodríguez M A V. Optimizing Multiple Sequence Alignment by Improving Mutation Operators of a Genetic Algorithm[J]. Intelligent Systems Design and Applications, 2009.ISDA '09. Ninth International Conference on, 2009:1257-1262.

[24] Chakrabarti T, Saha S, Sinha D. DNA Multiple Sequence Alignment by a Hidden Markov Model and Fuzzy Levenshtein Distance based Genetic Algorithm[J]. International Journal of Computer Applications, 2013: 73.

[25] Naznin F, Sarker R, Essam D. Progressive Alignment Method Using Genetic Algorithm for Multiple Sequence Alignment[J]. Evolutionary Computation, IEEE Transactions on, 2012, 16(5): 615-631.

[26] 张鑫源，胡晓敏，林盈. 遗传算法和粒子群优化算法的性能对比分析[J]. 计算机科学与探索，2014(1)：90-102.

[27] Zou Q, Shan X, Jiang Y. A Novel Center Star Multiple Sequence Alignment Algorithm Based on Affine Gap Penalty and K-Band[J]. Physics Procedia, 2012(33): 322-327.

[28] 杨锡南，孙啸. 生物信息学中基因数据可视化[J]. 计算机与应用化学，2001(18)：403-410.

[29] 贺向敏，周根宝. 基于遗传算法的多序列比对算法研究[D]. 内蒙古农业大学，2010.

[30] 张琎，张远. 基于 GC-GM 的多序列比对穷举遗传算法[J]. 计算机应用，2010，30(1)：146-149.

[31] 刘立芳，霍红卫，王宝树. PHGA-COFFEE：多序列比对问题的并行混合遗传算法求解[J]. 计算机学报，2006，29(5)：727-733.

[32] Fan H, Wu R, Liao B, et al. An Improved Genetic Algorithm for Multiple Sequence Alignment[J]. Journal of Computational and Theoretical Nanoscience, 2012, 9(10): 1558-1564.

[33] Thomsen R, Boomsma W. Multiple sequence alignment using SAGA: investigating the effects of operator scheduling, population seeding, and crossover operators[M]. Applications of Evolutionary Computing. Springer Berlin Heidelberg, 2004: 113-122.

[34] Lam T W, Sung W K, Tam S L, et al. Compressed indexing and local alignment of DNA[J]. Bioinformatics, 2008, 24(6): 791-797.

第 5 章　QPSO 算法在多序列比对中的应用

5.1　多序列比对的含义

生物信息学内涵非常丰富。其核心是基因组信息学，包括基因组信息的获取、处理、存储、分配和解释。基因组信息学的关键是“读懂”基因组的核苷酸顺序，即全部基因在染色体上的确切位置以及各 DNA 片段的功能；在发现新基因信息之后模拟和预测蛋白质空间结构，然后依据特定蛋白质的功能进行药物设计。生物序列中的信息在系统进化、生态守恒、疾病控制、病毒起源甚至 HIV 病毒统计和传播等的研究中是一个非常重要的基本工具。因此，序列比对是生物信息学的基础。

下面给出序列比对的定义：

生命最重要的物质基础是核酸(DNA 与 RNA)和蛋白质。DNA 分子中的核苷酸碱基分别是腺嘌呤(A)、鸟嘌呤(G)、胸腺嘧啶(T)、胞嘧啶(C)。因此，一般 DNA 分子看成是由字母表{A, G, T, C}中的元素组成的字母序列，而一切物种的蛋白质都是由 20 种氨基酸组成的，所以蛋白质也可以用 20 种氨基酸的单个字母{A, R, N, D, …, W, Y, V}组成的序列表示。给定包含 m 个长度不等序列的待比对序列集 $S=\left\{s^i \mid i=1,2,\cdots,m\right\}$，其中第 i 个序列 $s^i=(c_1^i,c_2^i,\ \cdots,c_l^i)$，$l$ 为序列 s^i 的长度，$c_j^i \in alph_set, j=1,2,\cdots,l$ 。$alph_set$ 为特定的可见字母

集合，c_j^i 表示一个核苷酸或氨基酸碱基。多序列比对就是通过在这些待比对序列中进行空位字符“–”的插入和删除操作，得到一个多序列比对结果的矩阵，其中。矩阵 A 中的每一列为一个位点上的比对，矩阵 A 的第 i 行对应参与比对的第 i 个序列，序列中非空字符的先后顺序在比对中保持不变。多序列比对的目标是使在比对结果中有尽可能多的列由相同的非空字符组成，同时在由不同字符组成的列中某一个或几个非空字符的数目尽可能多，以便发现不同序列之间的相似部分，进而推断它们在功能和结构上的相似性。对不对结果的评价可以采用不同形式的目标函数，求最佳比对的问题是一个 NP 完全问题。序列比对的根本任务是：通过比较生物分子序列，发现它们的相似性，找出序列之间共同的区域，同时辨别序列之间的差异。在分子生物学中，DNA 或蛋白质的相似性是多方面的，可能是结构的相似，可能是功能的相似，也可能是核酸或氨基酸序列的相似。一个较为普遍的规律是序列决定结构，结构决定功能。研究序列相似性的目的之一是通过相似的序列得到相似的结构或相似的功能。

序列比对分为全局比对和局部比对。全局比对要求把一个序列中的所有符号和另一个序列中的所有符号进行匹配比较，它描述整个序列的相似性；局部比对着眼于序列中的某些特殊片段，比较这些片段之间的相似性。将两个序列进行比对就是双序列比对，现在的标准算法是 1970 年由 Needleman 和 Wunsch 提出的基于动态规划方法的双序列比对算法。随着生物医学中有更多的序列合成出来，人们开始用多序列比对来更好地研究生物序列。将多个序列进行比对就是多序列比对问题。多序列比对问题是一个将不等长的多个序列通过插入空位变成等长的过程，这些位置上的空位代表着相比对的序列从共同的祖先通过插入/删除操作的进化过程。利用多序列比对算法得到的最优比对，可用于找出蛋白质家族的模体(motifs)或保守区域(conserved domains)；可用于预测蛋白质的结构和功能；可用于进行系统发育分析。目前主要有三种策略用于多序列比对：

第一种策略是“渐进比对”策略，其基本思想是：迭代地利用两序列动态规划算法，先由两条序列的比对开始，逐渐添加新序列，直到所有序列都加入为止。第二种策略是使用随机优化算法，根据目标函数值(通常为序列的得分函数)找出空位的最优位置，使序列比对的结果最优，如模拟退火算法(SA)、遗传算法(GA)。第三种策略基于概率模型的隐马尔可夫模型，使用 Baum-Welch 算法和 Viterbi 算法进行序列比对。下面使用的是第二种多序列比对策略，使用二进制的粒子群优化算法(BPSO)和二进制的量子粒子群优化算法(BQPSO)，并在这两种算法的过程中加入变异算子，避免比对的过程过早收敛，分别把这两种比对算法称为 MBPSO 算法和 MBQPSO 算法。

5.2　基于二进制 QPSO 算法的序列比对

5.2.1　二进制的 PSO 算法(BPSO)

二进制编码作为一种比较重要的编码形式，首先由 J.Kennedy 和 Eberhart 在 1997 年将基本微粒群算法应用于二进制编码，并做了大量的数值研究。

在二进制编码中，$x_j(t)$ 应取 0 或 1，在第 3 章介绍的 PSO 算法中，$v_j(t)$ 计算结果可能不是整数，且迭代后 $x_{j+1}(t)$ 不可能总是 0 或 1。

为此，Kennedy 引入模糊函数 $Sig(x)$，其定义为

$$Sig(x)=\frac{1}{1+\exp(-x)} \tag{5.1}$$

这样，PSO 算法中的迭代公式就变为

$$X_{id}=\begin{cases}0 & rand()\geqslant S(V_{id})\\ 1 & rand()<(V_{id})\end{cases} \tag{5.2}$$

需要说明的是：在基本微粒群算法中，$v_j(t)$ 表示速度，能够对当前位置 $x_j(t)$ 的方向和位置随机产生一定的影响，使得算法在给定区域上进行搜索。而在二进制编码的微粒群算法中，$v_j(t)$ 仅表示一个概率，即微粒的每一维分量的取值以 $Sig(v_j(t))$ 的概率取 1，而以 $1-Sig(v_j(t))$ 的概率取 0。

5.2.2　二进制的 QPSO 算法(BQPSO)

QPSO 算法的进化方程与 PSO 算法大相径庭，所以二进制 PSO 算法的一些定义和改进方法对于设计二进制 QPSO 算法是不适合的。在 QPSO 中，没有速度向量，只有位置向量和距离。由于粒子的位置被定义为一个二进制串，对距离和位置变换的定义就成了设计二进制 QPSO(binary QPSO，BQPSO)算法的第一步。

在所提出的 BQPSO 中，距离定义为两个二进制串的海明距离，即

$$|x-y| = d_H(x,y) \tag{5.3}$$

式中，x，y 是两个二进制串，也代表了两个位置向量；函数 $d_H()$ 表示求 x 和 y 的海明距离。

海明距离定义为两个串中不同位的个数。图 5.1 所示粒子 X_1 和粒子 X_2 的海明距离是 7。

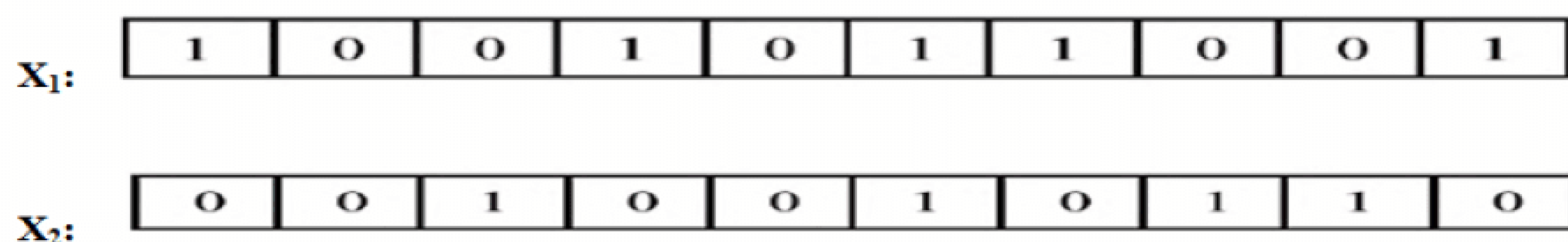


图 5.1　粒子 X_1 和粒子 X_2 的海明距离

设计 BQPSO 的关键问题是如何使 QPSO 算法的粒子进化方程转变成离散二进制空间中的操作。在 QPSO 中，平均最好位置(mbest)C 是通过求个体最好位置(pbest)的均值获得，在 BQPSO 中，C 的第 j 位取值由所有粒子的第 j 位的值确定。如果在第 j 位取 1 的粒子比取 0 的多，C 的第 j 位的值取 1，否则取 0。如果取 0 的粒子

数与取 1 的粒子数相等，则 C 在该位等概率地取 1 或 0，如图 5.2 所示。

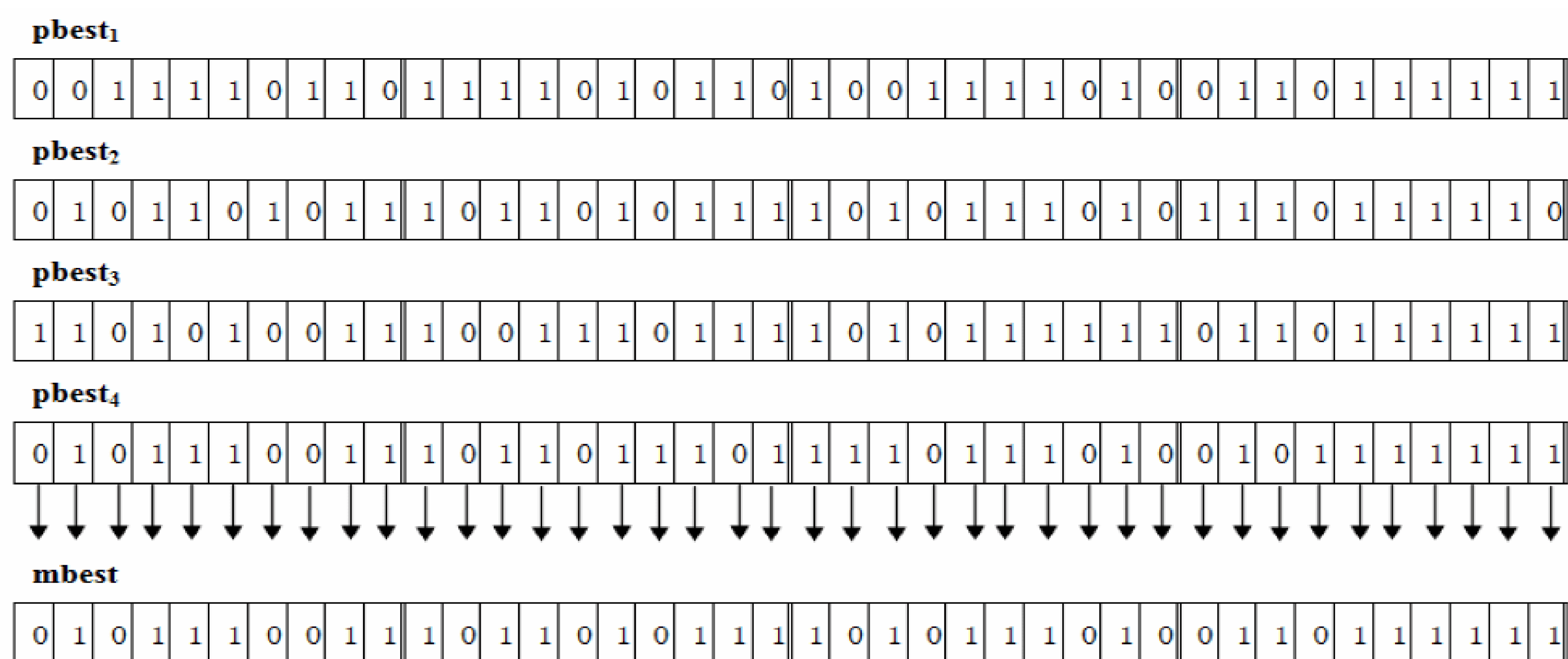


图 5.2　平均最好位置(mbest)

求平均最好位置的伪代码描述如下：

```
Get_mbest(y)(输入为所有粒子的个体最好位置)
for j=1 to N (the length of binary string)
    sum=0;
for 每一个粒子 i
        sum=sum+y[i][j];
endfor
    avg=sum/M;
if avg>0.5   C[j]=1;  endif
if avg<0.5   C[j]=0;  endif
if avg=0.5
if rand( )<0.5 C[j]=0;
else C[j]=1;
            endif
        endif
endfor
Return C
```

在以上伪代码中，函数 Get_mbest( )的输入是所有粒子的个体最好位置的二进制串，输出是代表平均最好位置 C 的二进制串。

在 QPSO 中，更新粒子的位置之前，必须求出局部吸引子 p_i，该点每一维坐标位于个体最好位置和全局最好位置在该维的坐标之间。于是 $p_i=(p_{i,1},p_{i,2},\cdots,p_{i,j})$ 就均匀地分布于以 P_i 和 G 为对角点的超矩形中。这样点 p_i 到 P_i 或 G 的距离必定都小于对角线的长度(P_i 和 G 的距离)，即

$$|p_i-P_i|\leqslant|y_i-G|，\quad|p_i-G|\leqslant|P_i-G| \tag{5.4}$$

可以推断当粒子的当前位置和个体最好位置向 p_i 点收敛时，群体的多样性不断减小，这对应着粒子的局部搜索。在 BQPSO 算法中， p_i 可以通过类似遗传算法中的交叉操作获得，即通过 P_i 和 G 的交叉产生两个子代，随机地选择一个子代作为 p_i 点。很明显，从海明距离的角度看，通过交叉产生的 p_i 满足式(5.4)。因此用单点或多点交叉产生 p_i 是合乎逻辑的。图 5.3 所示为通过多点交叉得到的局部吸引子 p_i。

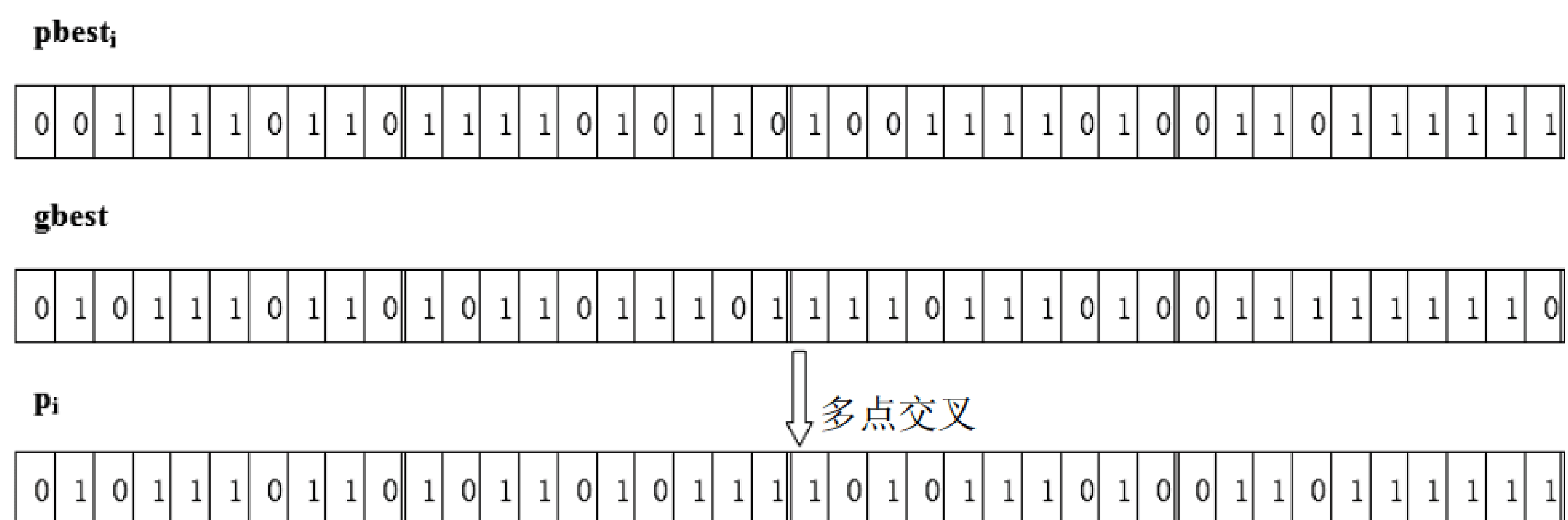


图 5.3　通过多点交叉得到的局部吸引子 p_i

以下是产生 p_i 点过程的描述。

Get_P(P_i ,G)
对 P_i 和 G 施加交叉操作产生两个子代个体 z_1 和 z_2;
if rand( )<0.5
p_i=z_1

```
else pi=z2;
endif
Return pi
```

考虑 QPSO 的粒子位置进化公式并把它改写为如下形式：

$$\left|X_{i,j}-p_{i,j}\right|=\alpha\left|C_j-X_{i,j}\right|\ln[1/u_{i,j}(t)],\ u_{i,j}(t)=Rand() \tag{5.5}$$

将距离看作海明距离，上式可写为

$$d_H[X_{i,j}(t+1),p_{i,j}(t)]=b \tag{5.6}$$

$$b=\alpha\bullet d_H[X_{i,j}(t)-C_j(t)]\bullet\ln[1/u_{i,j}(t)],\ u_{i,j}(t)=Rand() \tag{5.7}$$

式中，$d_H[X_{i,j}(t+1),p_{i,j}(t)]$ 为第 j 维的海明距离，它取 0 或 1。但当 b 的值大于 1 时，取海明距离为 1；当 $0<b<1$ 时，则可以产生一个随机数 *rand*()，当 *rand*()<b 时，距离取 1，否则取 0。从变异操作的角度，可将 b 看作每一位的变异概率 p_m，即

$$P_m=\begin{cases}b & b<1\\ 1 & 0<b<1\end{cases} \tag{5.8}$$

这样粒子位置的变化操作可由下面的伪代码描述：

```
Transf(pi, j, Pm)
for 对于 pi 中的每一个二进制位；
if rand()<Pm
if 该位的值为 1
置该位为 0;
else 否则置为 1;
endif
endif
endfor
Xi, j=pi, j;
Return Xi, j
```

有了以上的定义，BQPSO 算法的设计就基本完成，算法的描述如下：

(1) 初始化粒子的位置，即随机产生粒子的当前位置 X_i，并将

个体最好位置 P_i 置为 $P_i = X_i$。

(2) 调用函数 Get_mbest ( )，计算粒子群的平均最好位置 C。

(3) 将粒子当前位置 X_i 解码并评价粒子的目标函数值 $f(X_i)$(适应度值)，并与 $f(P_i)$ 的目标函数值进行比较，如果 $f(X_i) < f(P_i)$，则置 $P_i = X_i$。

(4) 找出群体的全局最好位置，即先求 $g = \arg\min\limits_{1 \leqslant i \leqslant M}(f(P_i))$，令 $G = P_g$。

(5) 对于每一个粒子，求 p_i，即 p_i =Get_P(P_i , G)。

(6) 由式(5.6)～(5.8)，求出 p_m。

(7) 根据 p_m，调用函数 Transf( )，得到粒子的新位置。

(8) 重复步骤(2)～(7)，直到满足停止准则或达到给定的最大代数。

5.2.3　基于 BPSO 或 BQPSO 的多序列比对

1. 种群编码

假设有 n 条序列进行比对，这 n 条序列的长度分别是 $l_1 \sim l_n$。本节中一个粒子代表了一个比对结果，这个比对结果以矩阵的方式进行存储。这个矩阵的每一行代表了一条比对的序列，每一行的最大长度为 $w = \lceil 1.2 \times l_{\max} \rceil$，这里 $l_{\max} = \max(l_1, l_2, \cdots, l_n)$。选择 1.2 作为乘数因子是基于观察一般多序列比对的结果，其空位数很少超过 20%。

2. 种群初始化

种群 P 的初始化就是用下列的方法不断地随机初始矩阵每一行。用随机数产生器产生从 $l_{\max}$ 到 $1.2 \times l_{\max}$ 的随机数作为每个个体长度，并根据长度随机产生插入空位的点的集合，长度超过此值的个体则舍弃。假设每个粒子的长度为 l_p，空位的位置在[0, l_p]之间随机产生。这行中剩余的位置插入对应的源序列。在 MBPSO 或 MBQPSO 算法中，“0”代表空位的位置，“1”代表源序列中字母的位置。

例如，对图 5.4 所示的三条源序列 S1、S2、S3 进行比对。

S1:　AGQYHECK
S2:　AFGPWERKYV
S3:　ASWIELKV

图 5.4　未比对的三条源序列

假设随机产生的粒子的长度是 12，随机初始的 S1 的空位的位置为(1, 1, 8, 8)，S2 的空位的位置为(2, 5)，S3 的空位的位置为(3, 3, 4, 7)。图 5.5 表示出了上述三条序列的二进制编码。

S1	1	0	0	1	1	1	1	1	1	1	0	0
S2	1	1	0	1	1	1	0	1	1	1	1	1
S3	1	1	1	0	0	1	0	1	1	1	0	1

图 5.5　三条已比对序列的二进制编码

根据上面的二进制编码可以得到相应的多序列比对结果，如图 5.6 所示。

S1	A	—	—	G	Q	Y	H	E	C	K	—	—
S2	A	F	—	G	P	W	—	E	R	K	Y	V
S3	A	S	W	—	—	I	—	E	L	K	—	V

图 5.6　三条已比对的序列

3. 适应值函数

为了证明算法的性能，使用两类打分函数来比较比对结果的好坏。

第一类打分函数为：在实验数据中不考虑参考比对，使用标准的 Sum-of-Pairs 分数来评价算法的性能，即

$$SOP = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} D(l_i, l_j) \tag{5.9}$$

这里是第 i 条已比对的序列，D 为距离矩阵：

(1) 对于核酸序列，使用 IUB 矩阵作为距离矩阵，IUB 矩阵是比较核酸序列的默认的分值矩阵，如果两条序列的两个残基是比对的，则分值为 1.9，反之，分值为 0。

(2) 对于蛋白质序列，使用 BLOSUM62 替换矩阵作为距离矩阵，这个矩阵似乎是最好可获得的数据库中执行相似性(同源性)搜索的。

为了避免已比对的序列中一条序列空位的积聚，从 SOP 分数中推演出仿射几何学的空位代价，对于比对结果中一条序列的空位代价按照下面的公式进行计算：

$$Gap\ \ \mathbf{cost} = GOP + n \times GEP \tag{5.10}$$

式中，GOP 表示第一个起始空位的固定罚分；GEP 表示对于扩展的空位的罚分，n 为一条序列中空位的个数。

对于已比对的每条序列的空位都要计算相应的空位代价。多序列比对结果的 SOP 的分值减去空位代价的总和，即为 SOP 的分值：

$$SOP = SOP - \sum_{i=1}^{N} Gap\ \mathbf{cost} \tag{5.11}$$

式中，N 为序列的个数。

GOP 和 GEP 的值分别设置为 11 和 2。

第二类打分函数为：在实验结果中，考虑参考比对，则使用如下的两个打分函数，这两个打分函数都使用了 BAliBASE 参考比对。

第一个函数是 Sum-of-Pairs Score(SPS)：设有 N 个比对测试序列，构成 M 列，标记第 i 列的比对列为 $A_{i1}, A_{i2}, \cdots, A_{iN}$。对每一对残基 A_{ij} 和 A_{ik}，定义变量 p_{ijk}，如果 A_{ij} 和 A_{ik} 在比对的参考结果中也位于同一列，则 $p_{ijk} = 1$，否则 $p_{ijk} = 0$。定义变量 S_i 为第 i 列的得分，

则有

$$S_i = \sum_{j=1, j \neq k}^{N} \sum_{k=1}^{N} p_{ijk} \tag{5.12}$$

记 SPS 为最终比对结果的得分，则

$$SPS = \sum_{i=1}^{M} S_i \Big/ \sum_{i=1}^{M_r} S_{ri} \tag{5.13}$$

式中，M_r 是参考比对结果中的列数；S_{ri} 是参考比对结果中第 i 列的得分。

用该函数进行评价时，比对结果的得分越多，说明比对的结果越好。

第二个函数是 Column Score(CS)：对比对结果的第 i 列，如果这列中所有的残基在参考比对中也位于同一列，则 $C_i = 1$，否则 $C_i = 0$。按照下面的公式对比对结果进行 CS 打分。

$$CS = \sum_{i=1}^{M} C_i / M_r \tag{5.14}$$

式中，M_r 是参考比对结果中比对列的个数。

4. 变异算子

为了避免在算法运行过程中粒子群出现早熟现象，在二进制的 PSO 或 QPSO 算法中加入变异算子，执行的操作为在全局最优位置 gbest 的位置向量上加入变异操作。具体的过程如下：

(1) 复制 gbest 向量，记 gbest_xin；

(2) 对每一个 gbest_xin，随机选取变异的位置 j，对每一个 gbest_xin 指定不同的变异的位置。

(3) 对每一个 gbest_xin，改变它的第 j 维的值，第 j 维如果为 0，就变为 1；如果为 1，就变为 0。

假如 f(gbest_xin)的值优于 f(gbest)的值，则在下一代的评估中用 gbest_xin 替换 gbest；假如 f(gbest)的值优于 f(gbest_xin)的值，则在

种群中删除 gbest_xin。

5. 程序流程

```
procedure MSA_MBPSO (MSA_MBQPSO)
being
初始化种群 P
While (没达到终止条件) do
Begin
For 每一个粒子 do
把二进制编码转换成多序列比对；
根据式(5.11)或(5.13)或(5.14)计算每一个粒子(每一个比对)的目标函数值；
end
在整个粒子群中找出全局最优位置：gbest=pbest[g][:]；
应用变异操作；
使用 BPSO 或 BQPSO 算法更新粒子的位置；
end
end
end
```

上面的程序运行结果会出现一个问题：根据长度，由粒子群优化算法速度公式产生的插入空位点的集合与实际需要空位数不一致的问题。解决办法：比实际需要的空位数多的情况，随机选取其中与实际需要空位数个数的点；比实际需要的空位数少的情况，通过随机数产生器产生所缺个数$\lceil 0,\lceil 1.2\times l_{\max}-l_i\rceil\rceil$个之间的整数，作为新增空位点添加到$v'_{id}$中。

图 5.7(a)和(b)分别给出了蛋白质序列家族 1idy 的 BPSO 和 BQPSO 的序列比对的结果；图 5.8(a)和(b)分别给出了蛋白质序列家族 1krn 的 BPSO 和 BQPSO 的序列比对的结果；图 5.9(a)和(b)分别给出了蛋白质序列家族 kinase 的 BPSO 和 BQPSO 的序列比对的结果。从 6 个序列比对的结果中可以得出，BQPSO 找到的相似性区域最多。

```
1idy            -------MEVKKTSWTEEEDRILYQAHKRLGNRWAEIAKLLPGR----TDNAIKNHWNST
1hstA           ----------SHPTYSEMIAAAIRAEKSRGGSSRQSIQKYIKSHYKVGHNADLQIKLS--
1aoy            -------MR-SSAKQEELVKAFKALLKEEKFSSQGEIVAALQEQ---GFDNINQSKVSRM
1jhgA           -------TPDEREALGTRVRIIEELLRGEMSQRE------LKNELGAGIATITRGSNS--
1tc3C           RGSALSDTERAQLDVMKLLNVSLHEMSRKISRSRHCIRVYLKDPVSYG------------
                                            .           :

1idy            MRR----KV
1hstA           IRRLLAAGV
1aoy            LTKFGAVRT
1jhgA           ---LKAAPV
1tc3C           --------T
                        .
```

(a) BPSO 序列比对的结果

```
1idy            MEVKKTSWTEEEDRILYQA--HKRL-GNRWAEIAKLLPG------R--TDNAIKNHWNST
1hstA           S---HPT-YSEMIAAAIRA--EKSRGGSSRQSIQKYIKSHYKVGHN--ADLQIKLSIRRL
1aoy            M---RSSAKQEELVKAFKAL-LKEEKFSSQGEIVAALQE---QGFDNINQSKVSRMLTKF
1jhgA           T----PD-EREALGTRVRIIEELLRGEMSQRELKNELG-------A--GIATITRGSNSL
1tc3C           R---GSA-LSDTERAQLDV--MKLL-NVSLHEMSRKIS-------R--SRHCIRVYLKDP
                     .    :                     .:   :               :

1idy            MRRKV
1hstA           LAAGV
1aoy            GAVRT
1jhgA           KAAPV
1tc3C           VSYGT
                    .
```

(b) BQPSO 序列比对的结果

图 5.7　蛋白质序列家族 1idy 的 BPSO 和 BQPSO 的序列比对的结果

```
plmn_petma      ACVKGTGEGYRGTAALTVSGKACQAWASQ-TPGDVYSCQGLVS--------NYCRNPDGE
urot_rat        DCYVGKGVTYRGTHSFTTSKASCLPWNSMILIGKTYTAWRANSQALGLGRHNYCRNPDGD
hgfa_human      -CFLGNGTGYRGVASTSASGLSCLAWNSDLLYQELHVDSVGAAALLGLGPHAYCRNPDND
fa12_human      SCYDGRGLSYRGLARTTLSGAPCQPWASEATYRNVTAEQ---ARNWGLGGHAFCRNPDND
urtg_desro      TCYKDQGVTYRGTWSTSESGAQCINWNSNLLIRRTYNGRMPEAVKLGLGNHNYCRNPDGA
                 *  . *  ***    : *   *  * *              :         :*****.

plmn_petma      KLPWCY-------TTEYCNVPS
urot_rat        AKPWCHVMKDRKLTWEYCDMSP
hgfa_human      ERPWCYVVKDSALSWEYCRLEA
fa12_human      IRPWCFVLNRDRLSWEYCDLAQ
urtg_desro      SKPWCYVIKARKFTSESCSVPV
                  ***.       : * * :
```

(a) BPSO 序列比对的结果

图 5.8　蛋白质序列家族 1krn 的 BPSO 和 BQPSO 的序列比对的结果

```
plmn_petma     ACVKGTGEGYRGTAALTVSGKACQAWASQTP-GDVYS-------CQGL-VSNYCRNPDGE
urot_rat       DCYVGKGVTYRGTHSFTTSKASCLPWNSMILIGKTYTAWRANSQALGLGRHNYCRNPDGD
hgfa_human     -CFLGNGTGYRGVASTSASGLSCLAWNSDLLYQELHVDSVGAAALLGLGPHAYCRNPDND
fa12_human     SCYDGRGLSYRGLARTTLSGAPCQPWASEATYRNVTA---EQARNWGLGGHAFCRNPDND
urtg_desro     TCYKDQGVTYRGTWSTSESGAQCINWNSNLLIRRTYNGRMPEAVKLGLGNHNYCRNPDGA
                *  . *  ***    : *   *  * *                  **    :*****.

plmn_petma     KLPWCY-------TTEYCNVPS
urot_rat       AKPWCHVMKDRKLTWEYCDMSP
hgfa_human     ERPWCYVVKDSALSWEYCRLEA
fa12_human     IRPWCFVLNRDRLSWEYCDLAQ
urtg_desro     SKPWCYVIKARKFTSESCSVPV
                 ***.       : * * :
```

(b) BQPSO 序列比对的结果

图 5.8　蛋白质序列家族 1krn 的 BPSO 和 BQPSO 的序列比对的结果(续)

```
kcc2_yeast     NYIFGRTLGAGSFGVVRQARKLSTNEDVAIKILLKKALQGNNVQL--QMLYEELSILQK-
daf1_caeel     QIRLTGRVGSGRFGNV--SRGDYRGEAVAVKVF-------NALDE--PAFHKETEIFETR
kpro_maize     TRKFKVELGRGESGTV-YKGVLEDDRHVAVKKL-------ENVRQGKEVFQAELSVIGR-
1csn           HYKVGRRIGEGSFGVIFEGTNLLNNQQVAIKF--------EPRRSDAPQLRDEYRTYKL-
dmk_human      DFEILKVIGRGAFSEVAVVKMKQTGQVYAMKIMNKWD---MLKRGEVSCFREERDVL---
                  .   :* *  . :        ..  *:*                  :  *

kcc2_yeast     -LSH----PNIVSF--KDWFESK--DKFYIVTQLATGGELFDRILSRGKFT----E----
daf1_caeel     MLRH----PNVLRYIGSDRVDTGFVTELWLVTEYHPSGSLHDFLLENTVNI----ETYYN
kpro_maize     -INH----MNLVRI--WGFCSEG--SHRLLVSEYVENGSLANILFSEGGNILLDWEGRFN
1csn           -LAGCTGIPNVYYF-----GQEG--LHNVLVIDLL-GPSLEDLLDLCGRKF---------
dmk_human      -VNGDRRWITQLHFAFQDE------NYLYLVMEYYVGGDLLTLLSKFGERI---------
                :       .                   :* :   . .*   :

kcc2_yeast     -----VDAVEIIVQILGAVEYMHSKNVVHRDLKPENVLY--VDKSENSPLVIADFG--IA
daf1_caeel     LMRSTASGLAFLHNQIGGSKESNKPAMAHRDIKSKNIMV-----KNDLTCAIGDLGLSLS
kpro_maize     IALGVAKGLAYLHHEC-------LEWVIHCDVKPENILL-----DQAFEPKITDFGLVKL
1csn           ---SVKTVAMAAKQMLARVQSIHEKSLVYRDIKPDNFLIGRPNSKNANMIYVVDFGMVKF
dmk_human      ---PAEMARFYLAEIVMAIDSVHRLGYVHRDIKPDNILL-----DRCGHIRLADFGSCLK
                            .              : *:*..*.:      ..     : *:*

kcc2_yeast     KQLKGEEDLI----YK-AAGSLGYVAPEVL------TQDGHGKP-CDIWSIGVITYTLLC
daf1_caeel     KPEDAASDIIANENYK--CGTVRYLAPEILNSTMQFTVFESYQC-ADVYSFSLVMWETLC
kpro_maize     LNRGGSTQNV----SH-VRGTLGYIAPEWV------SSLPITAK-VDVYSYGVVLLELLT
1csn           YRDPVTKQHIPYREKKNLSGTARYMSINTH------LGREQSRR-DDLEALGHVFMYFLR
dmk_human      LRADGTVRSLV------AVGTPDYLSPEILQAVGGGPGTGSYGPECDWWALGVFAYEMFY
                        :         *:  *:: :                  *  : . .    :

kcc2_yeast     GYS-----PFIAESVEGFMEECTASRYPVTFHMPYWDNISIDVKRF-ILKALRLNPADRP
daf1_caeel     RCEDGDVLPREAATVIPYIE--------------WTDRDPQDAQMFDVVCTRRLRPTENP
kpro_maize     G-------TRVSELVGG-----------------TDEVHSMLRKLVRMLSAKLEGEEQS
1csn           G----------SLPWQGLKAATNKQKYERIGEKKQSTPLRELCAGFPEEFYKYMHYARNL
dmk_human      G-------------QTPFYADSTAETYGKIVHYKEHLSLPLVDEGVPEEARDFIQRLLCP
                                                            .       :.

kcc2_yeast     TATELLDDP----------------WITSK
daf1_caeel     LWKDHPEMK----------------HIMEI
kpro_maize     WIDGYLDSKLNRPVNYVQARTLIKLAVSCL
1csn           AFDATPDYD-----------YLQGLFSKVL
dmk_human      -----PETRLGRGGAGDFRTHPFFFGLDWD
                     :
```

(a) BPSO 序列比对的结果

图 5.9　蛋白质序列家族 kinase 的 BPSO 和 BQPSO 的序列比对的结果

```
kcc2_yeast      NYIFGRTLGAGSFGVVRQARKLSTNEDVAIKILLKKALQGNNVQLQMLYEELSILQ--KL
daf1_caeel      QIRLTGRVGSGRFGNVSRGD--YRGEAVAVKVFNAL-------DEPAFHKETEIFETRML
kpro_maize      TRKFKVELGRGESGTVYKGV-LEDDRHVAVKKLENV-R----QGKEVFQAELSVIG--RI
1csn            HYKVGRRIGEGSFGVIFEGTNLLNNQQVAIKFEPRR------SDAPQLRDEYRTYK--LL
dmk_human       DFEILKVIGRGAFSEVAVVKMKQTGQVYAMKIMNKWDML-KRGEVSCFREERDVLV--NG
                   .   :* *  . :        ..  *:*                :  *

kcc2_yeast      S-HPNIVSFKDWFESK------DKFYIVTQLATGGELFDRILSRGKFTEVDA-V-E-IIV
daf1_caeel      R-HPNVLRYI--GSDRVDTGFVTELWLVTEYHPSGSLHDFLLENTVNIET----YYNLMR
kpro_maize      N-HMNLVRIWGFCSEG------SHRLLVSEYVENGSLANILFSEGGNILLDWEGRFNIAL
1csn            AGCTGIPNVYYFGQEG------LHNVLVIDL-LGPSLEDLLDLCGRKFSVKT-VAM-AAK
dmk_human       D-RRWITQLHFAFQDE------NYLYLVMEYYVGGDLLTLLSKFGERIPAEM-ARF-YLA
                     :       ..           :* :   . .*   :

kcc2_yeast      QILGAVEYM----------HSKNVVHRDLKPENVLYVDKS--EN-SPLVIADFGIAKQLK
daf1_caeel      STASGLAFLHNQIGGSKESNKPAMAHRDIKSKNIMVK-ND-----LTCAIGDLGLSLSKP
kpro_maize      GVAKGLAYLHHE-------CLEWVIHCDVKPENILLD-QA-----FEPKITDFGLVKLLN
1csn            QMLARVQSI----------HEKSLVYRDIKPDNFLIG-RPNSKNANMIYVVDFGMVKFYR
dmk_human       EIVMAIDSV----------HRLGYVHRDIKPDNILLD-RC-----GHIRLADFGSCLKLR
                     :  :                : *:*..*.:   .          : *:*

kcc2_yeast      G--EEDLI-----YKAAGSLGYVAPEVLTQDGH-------GKPCDIWSIGVITYTLLCGY
daf1_caeel      EDAASDIIA-NE-NYKCGTVRYLAPEILNSTMQFTVFES-YQCADVYSFSLVMWETLCRC
kpro_maize      RGGSTQNV-----SHVRGTLGYIAPEWVSSLPI-------TAKVDVYSYGVVLLELLTGT
1csn            DPVTKQHIPYREKKNLSGTARYMSINTHLGREQ-------SRRDDLEALGHVFMYFLRGS
dmk_human       ADGTVRSL------VAVGTPDYLSPEILQAVGGGPGTGSYGPECDWWALGVFAYEMFYGQ
                       :         *:  *:: :                  *  : .    :

kcc2_yeast      SPFIA---ESVEGFMEECTAS-RYPVTFHMPY--WD-NISIDVKRFI---LK-ALRLNPA
daf1_caeel      ED--G---D-----------VLPREAATVIPYIEWTDRDPQDAQMFD--VVC-TRRLRPT
kpro_maize      RV--S---ELVGGTDEVHSMLRKLVRMLSAKL-------EGEEQSWIDGYLD-SKLNRPV
1csn            LPWQGLKAATNK---QKYERIGEKKQSTPLRE--LCAGFPEEFYKYM----HYARNLAFD
dmk_human       TPFYA---DSTA---ETYGKIVHYKEHLSLPL--VDEGVPEEARDFI----Q--RLLCPP
                    .                                    :   :

kcc2_yeast      DRPT----A---TELLDDPWITSK
daf1_caeel      ENPL----W---KDHPEMKHIMEI
kpro_maize      NYVQ----A---RT-LIKLAVSCL
1csn            ATP-----D---YDYLQGLFSKVL
dmk_human       ETRLGRGGAGDFRTHPFFFGLDWD
```

(b) BQPSO 序列比对的结果

图 5.9　蛋白质序列家族 kinase 的 BPSO 和 BQPSO 的序列比对的结果(续)

5.3　本章小结

本章介绍了基于二进制的 PSO 和二进制的 QPSO 的多序列比对算法，为了避免算法的早熟，在算法过程中，加入了变异算子，从试

验结果中可以看出，不论对于核酸序列还是蛋白质序列，MBQPSO 得出的序列比对结果的 SOP、SPS、CS 的平均分值在绝大多数实例中都是最高的，因此 MBQPSO 算法的性能最优，MBPSO 的性能次之；在收敛速度上，虽然 MBQPSO 的收敛速度不如 SA、GA、SAGA、MBPSO 的收敛速度快，但是 MBQPSO 几乎在整个进化过程中，性能都优于 SA、GA、SAGA、MBPSO。但是 MBQPSO 和 MBPSO 算法，随着序列长度的增加，性能不如 CW、SAGA、T-Coffee 这些算法得出的序列比对的结果好。

在此需要指出的是，在 BPSO 和 BQPSO 中加入变异操作，增加了种群的多样性，可以加快种群的收敛速度，并且可以使算法避免陷入局部最优。在 GA 算法中，虽然也有许多交叉和变异操作，但是它们在实现多序列比对的过程中控制和操作起来非常复杂，SA 算法中冷却进度表的控制、衰减因子、迭代次数等参数的控制也非常复杂，并且 SA 对这些参数的设置都非常的敏感，在 SAGA 算法中，结合了 22 种不同的算子进行多序列比对，并且在两代之间进行变异操作。相对而言，MBPSO 和 MBQPSO 算法实现起来比较容易控制和操作。

还需要说明的是，在使用 SPS 或 CS 进行打分时，对于短序列和中序列来说，MBPSO 和 MBQPSO 取得了较好的分值，并且和参考比对相比找到了较多的相似区域。但是，对于长序列来说，MBPSO 和 MBQPSO 算法不如 CW、SAGA 和 T-Coffee 这些主要的多序列比对得到的得分高。原因是这些算法都是一些高度专业化的比对工具，它们在实现的过程中考虑到了方方面面的问题，例如在 T-Coffee 算法中，是基于渐进策略来避免在算法过程中出现的严重失误，这些失误是由算法的贪婪的本性造成的；在 SAGA 算法中，使用了 22 种不同的算子和变异操作来避免算法中出现异常情况，并且此算法在以 SOP 作为目标函数时，比其他算法的性能更好；在 CLUSTALW 实现的过程中，评估所有序列两两之间的进化距离来创建进化树，考虑所有序列之间的相关性。

总之，可以得出，MBPSO 和 MBQPSO 算法在多序列比对中是一个非常有效的优化算法。将来的研究工作的重心放在长序列上，设法改进 MBPSO 或 MBQPSO 算法，来提高比对的性能。

参 考 文 献

[1] Thompson JD, PlewniakF, Poch O. BAliBASE: A benchmark alignment database for the evaluation of multiple alignment programs [J]. Bioinformatics, 1999(15): 87-88.

[2] Thompson JD, PlewniakF, Poch O.A comprehensive comparison of multiple sequence alignment programs[J]. Nucleic Acids Res, 1999 (27): 2682-2690.

[3] Stoye J, Evers D, Meyer F. Rose: generating sequence families [J]. Bioinformatics, 1998(14): 157-163.

[4] JukesTH, Cantor CR. Evolution of protein molecules[C]. In: H. N. Munro, ed. Mammalian protein metabolism. New York: Academic Press, 1969: 21-123.

[5] Notredame C, Higgins D G.SAGA: sequence alignment by genetic algorithm[J]. Nuc Acids Res, 1996, 24(8):1515-1524.

[6] Gondro C, Kinghorn BP. A simple genetic algorithm for multiple sequence alignment[J]. Genetics and Molecular Research, 2007 (6): 964-982.

[7] ShiY, Eberhart R C. Empirical Study of Particle Swarm Optimization[C]. In: Proc 1999 Congress on Evolutionary Computation. Piscataway: NJ, 1999: 1945-1950.

[8] MamitsukaH. Finding the biologically optimal alignment of multiple sequences[J]. Artificial Intelligence in Medicine, 2005(35): 9-18.

第 6 章　基于隐马尔可夫模型和 QPSO 算法的多序列比对

6.1　引言

在上面的介绍中，已经知道目前主要有下列三种策略用于多序列比对，即“渐进比对”策略、使用随机优化算法和基于概率模型的隐马尔可夫模型。

下面的多序列比对过程使用的是第三种策略。在多序列比对的过程中，HMM 主要解决三个问题：一是得分问题，二是联配问题，三是训练问题。将得分问题用来评估模型的性能，联配问题用来实现多序列的比对，训练问题用来优化模型的参数。最常用的训练 HMM 模型的方法是基于统计和重估的方法，如 Baum-Welch 算法，但是 Baum-Welch 算法是一个局部最优算法，使用此算法得到的最终比对结果通常远离全局最优。最近还出现了粒子群算法(PSO)，此算法也是一个局部最优算法。

为了克服 Baum-Welch 算法和 PSO 算法的缺点，本章使用量子粒子群优化(QPSO)算法及其多样性控制的 QPSO 算法和多样性引导的 QPSO 算法来训练 HMM，不仅参数个数少，随机性强，并且能覆盖所有解空间，保证算法的全局收敛。

6.2　隐马尔可夫模型

隐马尔可夫模型(hidden markov model，HMM)是统计模型，它用来描述一个含有隐含未知参数的马尔可夫过程。其难点是从可观察的参数中确定该过程的隐含参数。然后利用这些参数来做进一步的分析，如模式识别。

在正常的马尔可夫模型中，状态对于观察者来说是直接可见的。这样状态的转换概率便是全部的参数。而在隐马尔可夫模型中，状态并不是直接可见的，但是受状态影响的某些变量则是可见的。每一个状态在可能输出的符号上都有一概率分布。因此输出符号的序列能够透露出状态序列的一些信息。

6.2.1　隐马尔可夫模型的基本原理

隐马尔可夫模型是在马尔可夫模型的基础上发展起来的。马尔可夫模型与隐马尔可夫模型的本质区别是隐马尔可夫模型观察到的符号并不是与状态一一对应，而是通过一组概率分布相联系。这样，站在观察者的角度，只能看到发出符号，不能直接看到状态。因此，不像马尔可夫模型观察到的符号和状态一一对应。

隐马尔可夫模型是一种统计模型，在语音识别等方面被广泛应用。隐马尔可夫模型也被较早地应用于生物信息学中的一些问题，如 DNA 编码区、蛋白质超家族(superfamily)的建模等。Rabiner 给出了隐马尔可夫模型的一个深入浅出的清晰介绍。一个隐马尔可夫模型可以由下列参数描述：

(1) N——模型中马尔可夫链的状态数目。记 N 个状态为 $\theta_1,\theta_2,\cdots,\theta_N$，那么状态空间表示为 $S=\{\theta_1,\theta_2,\cdots,\theta_N\}$。一般将状态空间简记为 $S=\{1,2,\cdots,N\}$。记 t 时刻马尔可夫链所处状态为 q_t，其中 $q_t\in S$。

(2) M——每个状态对应的可能的观察符号数目。记 M 个观察

符号为 $v_1, v_2, \cdots, v_M$，那么离散符号集(或称字母表)表示为 $V = \{v_1, v_2, \cdots, v_M\}$。记 t 时刻观察到的符号为 o_t，其中 $o_t \in V$。

(3) $\boldsymbol{\pi}$——初始状态概率向量。$\boldsymbol{\pi} = (\pi_1, \pi_2, \cdots, \pi_N)$，其元素 π_i 是 t=1 时(即初始时刻)处于状态 i 的概率，即 $\pi_i = P(q_{t+1} = j \mid q_t = i)$，$1 \leqslant i, j \leqslant N$。

(4) $\boldsymbol{A}$——状态转移概率矩阵。$\boldsymbol{A} = [a_{ij}]_{N \times N}$，其元素 a_{ij} 是指 t 时刻状态为 i 时，t+1 时刻状态为 j 的概率，即 $a_{ij} = P(q_{t+1} = j \mid q_t = i)$，$1 \leqslant i, j \leqslant N$。

(5) $\boldsymbol{B}$——符号发出概率矩阵。$\boldsymbol{B} = [b_j(k)]_{N \times M}$，其元素 $b_j(k)$ 是指 t 时刻状态为 j 时，输出观测符号 v_k 的概率，即 $b_j(k) = P(o_t = v_k \mid q_t = j)$，$1 \leqslant j \leqslant N$，$1 \leqslant k \leqslant M$。

这样，可以记一个隐马尔可夫模型为 $\lambda = (N, M, \boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$，或简写为 $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$。

6.2.2　隐马尔可夫模型的基本问题与算法

为了将隐马尔可夫模型应用于实际，必须解决三个关键的基本问题：

(1) 得分问题。给定隐马尔可夫模型和一条可观察的符号序列，欲知道给定隐马尔可夫模型产生该条可观察符号序列的概率。

(2) 联配问题。给定隐马尔可夫模型和一条可观察的符号序列，欲知道给定隐马尔可夫模型产生该条可观察符号序列的最可能的(或最佳的)状态序列。

(3) 训练问题。给定一条可观察的符号序列数据，欲找到最能说明该条序列数据的隐马尔可夫模型的构型和参数。

这三个问题可以分别使用向前算法或向后算法、Viterbi 动态规划算法和 Baum-Welch 重估计(EM)算法来求解。

为了使隐马尔可夫模型在数学上和计算上易于处理，需要对模型在理论上做如下的假设。

假设 6.1：对于可观察符号序列 $o_1 o_2 \cdots o_t$ 和状态序列 $q_1 q_2 \cdots q_t$，

假定有

$$P(o_t \mid q_1 q_2 \cdots q_t, o_1 o_2 \cdots o_{t-1}) = P(o_t \mid q_t) \tag{6.1}$$

式(6.1)说明，在隐马尔可夫模型中，时刻 t 输出的符号仅与此刻的状态有关，而与此前输出的符号和状态无关。

假设 6.2： 对于可观察符号序列 $o_1 o_2 \cdots o_t$ 和状态序列 $q_1 q_2 \cdots q_{t+1}$，假定有

$$P(q_{t+1} \mid q_1 q_2 \cdots q_t, o_1 o_2 \cdots o_t) = P(q_{t+1} \mid q_t) \tag{6.2}$$

式(6.2)说明，在隐马尔可夫模型中，时刻 t+1 的状态取值仅与时刻 t 的状态取值有关，而与此前输出的符号和状态无关。

1. 向前算法

定义向前变量 $a_t(i)$ 为

$$a_t(i) = P(o_1 o_2 \cdots o_t, q_t = i \mid \lambda)，\ 1 \leqslant t \leqslant T \tag{6.3}$$

式中，$a_t(i)$ 是给定模型 λ，在时刻 t 状态为 i 时观察到的部分序列 $o_1 o_2 \cdots o_t$ 的概率。

向前算法的具体实现步骤如下：

(1) 初始化：

$$\boldsymbol{a}_1(i) = \pi_i b_i(o_1)，\ 1 \leqslant i \leqslant N \tag{6.4}$$

(2) 递归计算：

$$a_{t+1}(j) = \left[\sum_{i=1}^{N} a_t(i) a_{ij}\right] b_j(o_{t+1}),\ 1 \leqslant t \leqslant T-1，\ 1 \leqslant j \leqslant N \tag{6.5}$$

(3) 最终结果：

$$P(O \mid \lambda) = \sum_{i=1}^{N} a_T(i) \tag{6.6}$$

2. 向后算法

定义向后变量 $\beta_t(i)$ 为

$$\beta_t(i) = P(o_{t+1} o_{t+2} \cdots o_T \mid q_t = i, \lambda)，\ 1 \leqslant t \leqslant T-1 \tag{6.7}$$

式中，$\beta_t(i)$ 是给定模型 λ，在 t 时刻状态为 i 时观察到的部分序列

$o_{t+1}o_{t+2}\cdots o_T$ 的概率。

向后算法的具体实现步骤如下：

(1) 初始化：

$$\beta_T(i)=1\text{，}\quad 1\leqslant i\leqslant N \tag{6.8}$$

(2) 递归计算：

$$\beta_t(i)=\sum_{j=1}^{N}a_{ij}b_j(o_{t+1})\beta_{t+1}(j)\text{，}\quad t=T-1,T-2,\cdots,1\text{，}\quad 1\leqslant i\leqslant N \tag{6.9}$$

(3) 最终结果：

$$P(O\mid\lambda)=\sum_{i=1}^{N}\pi_i b_i(o_1)\beta_1(i) \tag{6.10}$$

3. Viterbi 动态规划算法

详见 6.3.3 节。

4. Baum-Welch 重估计(EM)算法

为了实施 Baum-Welch 重估计(EM)算法，对于给定的训练序列 O 和模型 λ，定义 $r_t(i)$ 为时刻 t 时状态序列处于状态 i 的概率，即

$$r_t(i)=P(q_t=i\mid Q,\lambda) \tag{6.11}$$

利用隐马尔可夫模型假设和向前/向后算法，有

$$r_t(i)=\frac{P(q_t=i,O\mid\lambda)}{P(O\mid\lambda)}=\frac{a_t(i)\beta_t(i)}{P(O\mid\lambda)}=\frac{a_t(i)\beta_t(i)}{\sum_{j=1}^{N}a_t(j)\beta_t(j)} \tag{6.12}$$

类似地，对于给定的训练序列 O 和模型 λ，定义 $\xi_t(i,j)$ 为时刻 t 时状态序列处于状态 i 和时刻 t+1 时处于状态 j 的概率，即

$$\xi_t(i,j)=P(q_t=i,q_{t+1}=j\mid O,\lambda) \tag{6.13}$$

同样，利用隐马尔可夫模型假设和向前/向后算法，有

$$\xi_t(i,j)=\frac{P(q_t=i,q_{t+1}=j,O\mid\lambda)}{P(O\mid\lambda)}=\frac{a_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}a_t(j)\beta_t(j)} \tag{6.14}$$

根据 $r_t(i)$ 和 $\xi_t(i,j)$ 的定义，$r_t(i)$ 和 $\xi_t(i,j)$ 之间满足关系：

$$r_t(i)=\sum_{j=1}^{N}\xi_t(i,j) \tag{6.15}$$

6.3　基于剖面 HMM 和 QPSO 的多序列比对

本节使用的是一种标准的剖面 HMM 的拓扑结构用于多序列的比对，最初由 Krogh 等(1994)提出，如图 6.1 所示。该模型包含了一系列的状态 $(S_1,S_2,\cdots,S_n)$，这些状态被分成三组，分别是匹配状态(M)、插入状态(I)和缺失状态(D)。该模型是包含这三种状态重复集的简单的从左至右的结构。为了研究的方便，再引入两个额外的状态：开始状态(begin)和结束状态(end)。状态之间通过转移概率 a_{ij} 相联系，转移概率具有下面的性质：$a_{ij}\geqslant 0$，$1\leqslant i,j\leqslant n$ 并且 $\sum_{j=1}^{n}a_{ij}=1$，$1\leqslant i\leqslant n$。一个匹配状态或一个插入状态按照一定的符号发出概率 $b_j(k)$ 发出一个可见符号 v_k，符号发出概率具有下面的性质：$b_j(k)\geqslant 0$，$1\leqslant j\leqslant n$，$1\leqslant k\leqslant M$ 并且 $\sum_{k=1}^{M}b_j(k)=1$，$1\leqslant j\leqslant n$，这里 M 为每个符号对应的观察符号数目。缺失状态、开始状态、结束状态都不发出任何符号。

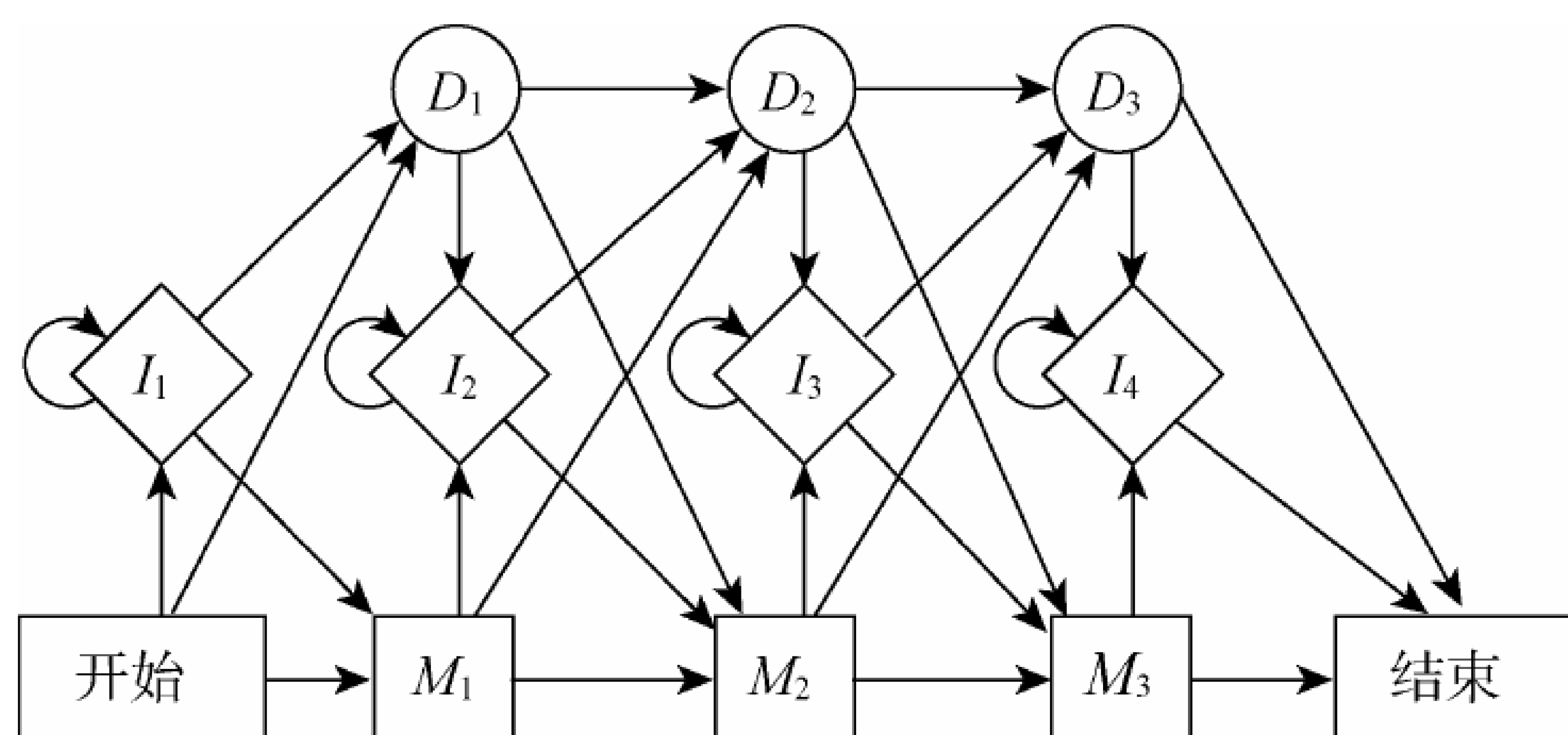


图 6.1　用于多序列比对的隐马尔可夫模型

当使用图 6.1 所示的 HMM 进行多序列比对时，每条序列从开始到结束通过这些状态穿越模型，在这些待比对序列中进行空位字符“－”的插入和删除操作，得到一个多序列比对结果的矩阵 $\boldsymbol{A}=(a_{ij})_{m\times n}$，其中 $a_{ij}\in alph_set\cup\{-\}$。矩阵 $\boldsymbol{A}$ 中的每一列为一个位点上的比对，矩阵 $\boldsymbol{A}$ 的第 i 行对应参与比对的第 i 个序列，序列中非空字符的先后顺序在比对中保持不变。多序列比对的目标是使在比对结果中有尽可能多的列由相同的非空字符组成，同时在由不同字符组成的列中某一个或几个非空字符的数目尽可能多，以便发现不同序列之间的相似部分，进而推断它们在功能和结构上的相似性。

用量子粒子群优化算法训练剖面 HMM 时，每一个粒子代表一个 HMM，通过不断地更新粒子的位置来优化 HMM。在训练中保持模型的长度不变，仅仅优化模型的参数：转移概率和符号发出概率。对图 6.1 所示的 HMM 的拓扑结构，取待比对序列的平均值 m 为模型的长度，不考虑初始状态和结束状态，则模型的状态数为 $3m+1$，状态转移概率参数为 $3(3m+1)$ 个；设字符集大小为 $|A|$，共有 $(2m+1)|A|$ 个符号发出概率。所以每个粒子是维数为 $9m+3+(2m+1)|A|$ 的一个实数编码串。所以 DNA 模型的参数个数是 $17m+7$，蛋白质模型的参数个数是 $49m+23$。根据转移概率和符号发出概率的性质，在对粒子对应的 HMM 模型进行评价前，需要先对 HMM 中的状态转移概率和符号发出概率进行归一化，以满足 $3m+1$ 个转移概率归一化约束方程和 $2m+1$ 个符号发出概率归一化方程。

根据粒子群算法训练的结果，得到全局最优的粒子对应的 HMM，接下来用此模型使用 Viterbi 算法进行序列的比对，得到最优的比对结果，并用基于 SP(sum of pairs)打分系统的目标函数评估比对的结果。

在整个算法的过程中，根据剖面 HMM 的拓扑结构，所使用的转移概率的形式如表 6.1 所示。

表 6.1　转移概率

Transition	*Transition*	*Transition*
$M_i \to M_{i+1}$	$I_i \to M_i$	$D_i \to M_{i+1}$
$M_i \to I_{i+1}$	$I_i \to I_i$	$D_i \to I_{i+1}$
$M_i \to D_{i+1}$	$I_i \to D_i$	$D_i \to D_{i+1}$

算法流程如图 6.2 所示。

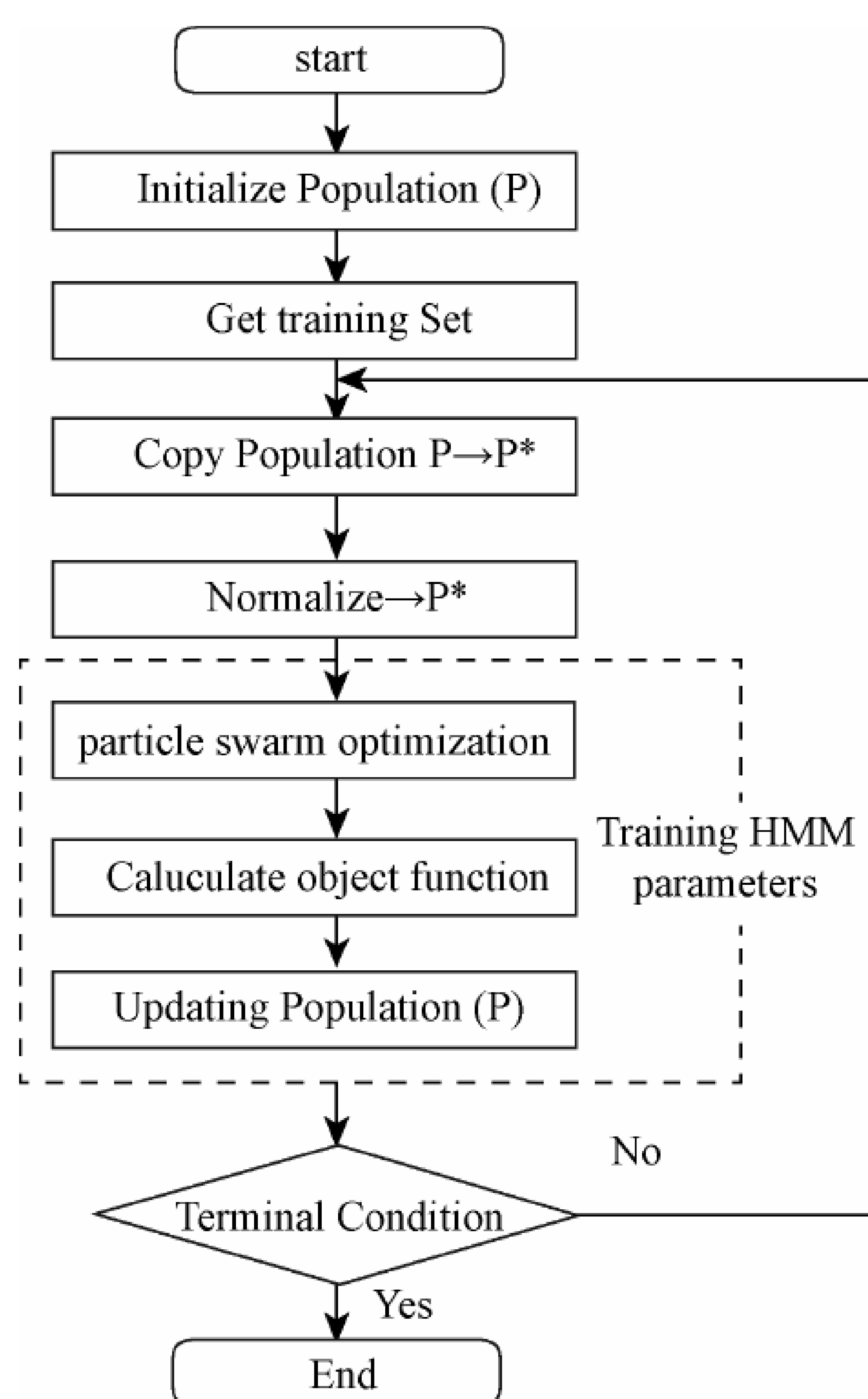


图 6.2　基于剖面 HMM 和 PSO 优化算法的多序列比对的算法流程

6.3.1　融合多样性的 QPSO 算法

1. 多样性的基本思想

众所周知，对于多峰函数的优化问题，PSO 算法和其他进化算

法遇到的最大问题就是早熟。早熟问题会导致算法的收敛性能下降和次优解的产生。在 PSO 中，由于粒子群的聚集性，粒子间会快速地交换信息，这样使得群体的多样性很快降低，一旦 PSO 算法落入局部最优解或次优解，就很难跳出来。于是粒子群的搜索就陷于停滞。QPSO 算法是全局收敛算法，粒子间的等待效应使算法的全局搜索能力大大提高。但是对于复杂问题，早熟问题也是难以避免的，这同样也是由于粒子群的聚集性和收敛导致多样性的降低引起的。

2002 年，Ursem 提出了一个多样性引导的进化算法(diversity-guided evolutionary algorithm，DGEA)。该方法是基于十进制遗传算法的。算法中使用了使多样性减小的算子——选择和交叉，以及使多样性增加的算子——变异。算法以个体到中心点的平均距离度量多样性。根据多样性的大小，使群体在多样性减小和多样性增加这两个模式中切换，使算法能够持久地进行搜索，直到发现最优解。实验表明，DGEA 算法在求解多峰函数优化问题时性能非常好。

同样在 2002 年，Riget 等在 PSO 中引入了 Ursem 控制群体多样性的思想，提出了吸引－排斥粒子群算法(attraction-repulsion Particle swarm optimization，ARPSO)。ARPSO 算法通过控制粒子群的多样性，使算法在全局搜索和局部勘探两个模式之间进行切换。这两种模式对应了两个阶段：吸引和排斥。其中吸引就是收敛阶段，这与普通的 PSO 算法没有两样。而排斥阶段就是使粒子反向运动，远离 p_i 点，这样使粒子间的距离扩大从而达到增加群体多样性的目的。

2. 多样性的度量

受 Ursem 和 Riget 的启发，在 QPSO 算法中引入多样性控制的方法。粒子群的多样性也采用粒子到中心点的平均欧几里的距离来度量。然而在 PSO 和 QPSO 中，存在两个群体，即由粒子的当前位置组成的群体和由粒子的个体最好位置组成的群体，它们分别表示为

$$S_X = (X_1, X_2, \cdots, X_M) \tag{6.16}$$

$$S_P = (P_1, P_2, \cdots, P_M) \tag{6.17}$$

相应的，QPSO 或 PSO 就由两种形式的多样性度量，即

$$diversity(S_X)=\frac{1}{M\cdot|A|}\cdot\sum_{i=1}^{M}\sqrt{\sum_{j=1}^{N}(X_{i,j}-\overline{X_j})^2},\quad \overline{X_j}=\frac{1}{M}\sum_{i=1}^{M}X_{i,j} \tag{6.18}$$

$$diversity(S_P)=\frac{1}{M\cdot|A|}\cdot\sum_{i=1}^{M}\sqrt{\sum_{j=1}^{N}(P_{i,j}-\overline{P_j})^2},\quad \overline{P_j}=\frac{1}{M}\sum_{i=1}^{M}P_{i,j}=C_j \tag{6.19}$$

式中，$|A|$ 是搜索空间最长对角线的长度；N 是优化问题的维数；M 是粒子群的规模。

3. 多样性控制的 QPSO 算法

算法的基本思想：QPSO 算法在搜索开始时，由于粒子群的初始化，多样性相对比较高。在随后的搜索过程中，由于粒子的逐渐收敛，群体的多样性不断下降，结果是算法的局部搜索能力不断地加强，而全局搜索能力不断减弱。在搜索的早期和中期，多样性的减小对于粒子群的搜索效率的提高是必需的。然而，到了搜索的后期，由于粒子都聚集到一个相对较小的区间，这时粒子群的多样性已经很低，全局搜索能力变得很弱，进行大范围搜索的可能性已经很小。此时，如果全局最好位置(gbest)位于局部最优解或次优解，算法就会发生早熟现象。

为了能有效地避免早熟现象，提高 QPSO 算法的性能，在本部分和下一部分提出了多样性控制的 QPSO 算法(diversity-controlled QPSO，DCQPSO)和多样性引导的 QPSO 算法(diversity-guided QPSO，DGQPSO)，且它们与 Ursem 和 Riget 的工作不同，在这两个算法中只对多样性设置下限 d_{low}。这里首先讨论 DCQPSO 算法。

在 DCQPSO 算法中，使用多样性测度 $diversity(S_X)$ 对算法的搜索进行引导。在粒子群初始化后，进入收敛状态。这时收缩－扩张系数从 1.0 到 0.5 线性地减小，即

$$\alpha=(1.0-0.5)\times(t_{\max}-t)/t_{\max}-0.5 \tag{6.20}$$

式中，$t_{\max}$ 是最大迭代数(即最大进化代数)；t 是当前迭代次数。

这种参数控制方法与 QPSO 算法相同。在收敛过程中，一旦多样性 $diversity(S_X)$ 小于预先设定的下限 d_{low}，使粒子群进入发散状态，这是多样性暂时性地增加，直到大于 d_{low}。

有两种方法使粒子进入发散状态。一种是通过控制参数。由第 3 章的粒子收敛分析可以知道，当 $\alpha < 1.781$ 时，粒子能收敛到局部吸引子 p_i；当 $\alpha > 1.781$ 时，粒子发散。因此，可以考虑当多样性降到 d_{low} 以下时，将设置为某个值，粒子群就会进入发散状态。而当粒子群的多样性达到 d_{low} 以上时，粒子重新进入收敛状态，这时的值仍旧根据变化。将这种 DCQPSO 称为 DCQPSO1。

另一种增加多样性的方法是初始化平均最好位置 C。当粒子群的多样性较小的时候，粒子的当前位置和 C 的距离很小，进化方程中的 L 变得很小，这样粒子远离吸引子 p_i 的概率也非常小。对 C 进行初始化，相当于对其实施变异，使之远离粒子群，这样 C 与粒子的当前位置拉开，粒子能远离 p_i 点，于是多样性就增加。使用该方法的 DCQPSO 称为 DCQPSO2。

DCQPSO 算法的流程如下：

(1) 初始化粒子群。

(2) 当迭代次数小于 $t_{\max}$ 时，执行以下步骤。

(3) 计算粒子群的平均最好位置 C。

(4) 计算粒子群的多样性 $diversity(S_X)$ 。

(5) 计算收缩—扩张系数：$\alpha = (1.0 - 0.5) \times (t_{\max} - t) / t_{\max} - 0.5$ (收敛模式)。

(6) 判断 $diversity(S_X)$ ，如果小于 d_{low}，则设置 $\alpha = \alpha_0$ (DCQPSO1)；或者，初始化平均最好位置 C(DCQPSO1)(发散模式)。

(7) 对每个粒子按照 QPSO 进化方程进行位置更新。

(8) 返回步骤(2)。

4. 多样性引导的 QPSO 算法

算法的基本思想：对于前面的 DCQPSO，使用并控制 $diversity(S_X)$

来引导 QPSO 的搜索。如果在算法中控制 $diversity(S_P)$ 来引导粒子群的搜索，性能很差，这主要是由 DCQPSO 中使粒子发散的方法所引起的。这里使用另外一种增加多样性的方法，这种方法使两种多样性度量都可用来引导 QPSO 的搜索。也就是，当粒子群的多样性 $diversity(S_X)$ 或 $diversity(S_P)$ 降低到 d_{low} 以下时，对粒子群中处于全局最好位置的粒子进行如下的变异操作：

$$P_{g,j} = P_{g,j} + \gamma \cdot |A| \cdot \varepsilon, \quad \varepsilon \sim N(0,1), \ (j = 1, 2, \cdots, N) \tag{6.21}$$

式中，ε 是服从标准正态分布的随机数；γ 是参数。

当变异操作施加到全局最好的粒子时，会使粒子个体最好位置到平均最好位置 C 的平均距离增加，从而使多样性 $diversity(S_P)$ 增加。同时，由于全局最好位置的变动会使 C 偏离目前的位置，粒子当前位置与 C 的距离也会扩大，这就导致粒子有一定程度的发散(因为粒子概率分布的方差变大)，从而使 $diversity(S_X)$ 增加。因此，使用变异方法可以同时增加两种多样性。针对这种方法，提出了两种多样性引导的 QPSO 算法(DGQPSO)，一种使用多样性 $diversity(S_X)$ 引导算法搜索，称为 $DGQPSO_X$；另外一种使用多样性 $diversity(S_P)$，称为 $DGQPSO_P$。DGQPSO 算法的流程如下：

(1) 初始化粒子群。

(2) 当迭代次数小于 t_{max} 时，执行以下步骤。

(3) 计算粒子群的平均最好位置 C。

(4) 计算粒子群的多样性 $diversity(S_X)$ ($DGQPSO_X$) 或 $diversity(S_P)$ ($DGQPSO_P$)。

(5) 计算收缩－扩张系数：$\alpha = (1.0 - 0.5) \times (t_{max} - t) / t_{max} - 0.5$ (收敛模式)。

(6) 判断 $diversity(S_X)$ 或 $diversity(S_P)$，如果小于 d_{low}，则对全局最好粒子实施式(6.21)的变异(发散模式)。

(7) 对每个粒子按照 QPSO 进化方程进行位置更新。

(8) 返回步骤(2)。

6.3.2　评估训练算法的质量

在粒子群优化算法中，需要评估每个粒子所代表的模型的质量。使用的评估函数为

$$Log_odds(O,\lambda) = -\frac{1}{N}\sum_{i=1}^{N}\frac{\log_2 P(O_S \mid \lambda)}{l_i} \tag{6.22}$$

式中，$O=\{O_1,O_2,\cdots,O_M\}$ 是给定的待比对序列的集合，序列的个数为 M 个；l_S 是序列 O_S 的长度的值越大，说明使用量子粒子群优化算法训练得到的 HMM 模型的参数是最优的，模型的稳定性和可靠性都较好，可以使用向前算法计算 *Log_odds* 的值。

6.3.3　模型的联配问题

联配问题可归结为给定可观察符号序列 $O=o_1o_2\cdots o_T$ 和已知的隐马尔可夫模型 $\lambda=(\boldsymbol{\pi},\boldsymbol{A},\boldsymbol{B})$，在最佳意义上确定一条状态序列 $Q^*=q_1^*q_2^*\cdots q_T^*$ 的问题。联配问题是力图揭露出模型中隐藏着的部分，即找出最好的状态序列。这里使用 Viterbi 算法进行序列的联配。

定义 Viterbi 变量：

$$\delta_t(i) = \max_{q_1q_2\cdots q_{t-1}} P(q_1q_2\cdots q_t, q_t = i, o_1o_2\cdots o_t \mid \lambda) \tag{6.23}$$

式中，$\delta_t(i)$ 是时刻 t 时沿一条路径 $q_1q_2\cdots q_t$，且 $q_t=i$，产生出可观察符号序列 $o_1o_2\cdots o_t$ 的最大概率。

$\delta_t(i)$ 可通过递归法进行计算，即

$$\delta_t(i) = \max_{1\leqslant j\leqslant N}[\delta_{t-1}(j)a_{ji}]b_i(o_t) \tag{6.24}$$

对于缺失状态由于没有字符生成，所以有

$$\delta_t(i) = \max_{1\leqslant j\leqslant N}[\delta_{t-1}(j)a_{ji}] \tag{6.25}$$

为了实际找到最佳状态，需要跟踪使式(6.25)最大的参数变化的轨迹(对每个 t 和 i 值)。这可以借助于定义 $\phi_t(i)$ 来标记 t 时刻的状态

i 最可能是由 $t-1$ 时刻的哪个状态转移而来。那么，寻找最佳状态序列 Q^* 的完整过程可陈述如下：

(1) 初始化

$$\delta_1(i)=\pi_i b_i(o_1),\ 1\leqslant i\leqslant N;\quad \phi_1(i)=0,\ 1\leqslant i\leqslant N \tag{6.26}$$

(2) 递归计算

$$\delta_t(i)=\max_{1\leqslant j\leqslant N}\left[\delta_{t-1}(j)a_{ji}\right]b_i(o_t),\quad 2\leqslant t\leqslant T,\ 1\leqslant i\leqslant N \tag{6.27}$$

$$\phi_t(i)=\arg\max_{1\leqslant j\leqslant N}\left[\delta_{t-1}(j)a_{ji}\right],\quad 2\leqslant t\leqslant T,\ 1\leqslant j\leqslant N \tag{6.28}$$

(3) 中断

$$P^*=\max_{1\leqslant i\leqslant N}\left[\delta_T(i)\right] \tag{6.29}$$

$$q_T^*=\arg\max_{1\leqslant i\leqslant N}\left[\delta_T(i)\right] \tag{6.30}$$

(4) 路径(最佳状态序列)回溯

$$q_t^*=\phi_{t+1}\left(q_{t+1}^*\right),\quad t=T-1,T-2,\cdots,1 \tag{6.31}$$

给定三条待比对序列：

O(1)——AGQYHECK；

O(2)——AFGPWERKYV；

O(3)——ASWIELKV。

按照下面的两个步骤可以得到序列比对的结果：

(1) 根据 Viterbi 算法和图 6.1 所示的 HMM 拓扑结构找出每条序列所对应的状态序列，如图 6.3 所示，每条序列的氨基酸碱基对应一个匹配状态或一个插入状态。

(2) 由图 6.3 生成的状态序列，可以进行空位字符“－”的插入操作，如图 6.4 所示。所有的序列中与匹配状态 M 相对应的氨基酸碱基是比对的，这些氨基酸碱基位于同一列；与插入状态 I_K 相对应的氨基酸碱基位于 M_{K-1} 或 D_{K-1} 之后， M_K 或 D_K 之前。

由步骤(1)和步骤(2)，得到联配后的序列，如图 6.5 所示。

Q*(1)	M_1	M_4	I_5	I_5	M_5	M_6	I_7	M_7		
O(1)	A	G	Q	Y	H	E	C	K		
Q*(2)	M_1	M_2	M_4	I_5	I_5	M_6	I_7	M_7	M_8	M_9
O(2)	A	F	G	P	W	E	R	K	Y	V
Q*(3)	M_1	M_2	I_3	I_5	M_6	I_7	M_7	M_9		
O(3)	A	S	W	I	E	L	K	V		

图 6.3　每条序列所对应的状态序列

Q*(1)	M_1	D_2	D_3	M_4	I_5	I_5	M_5	M_6	I_7	M_7	D_8	D_9
O(1)	A	—	—	G	Q	Y	H	E	C	K	—	—
Q*(2)	M_1	M_2	D_3	M_4	I_5	I_5	D_5	M_6	I_7	M_7	M_8	M_9
O(2)	A	F	—	G	P	W	—	E	R	K	Y	V
Q*(3)	M_1	M_2	I_3	D_3	D_4	I_5	D_5	M_6	I_7	M_7	D_8	M_9
O(3)	A	S	W	—	—	I	—	E	L	K	—	V

图 6.4　空位字符“—”的插入

```
O(1):
A —  — G  Q Y  H  E C K —    —
O(2):
A F  — G  P W  —  E R K Y    V
O(3):
A S  W —  — I  —  E L K —    V
```

图 6.5　联配后的序列

6.3.4　评估比对序列的质量

在利用 Viterbi 算法获得的路径进行比对后，需要通过基于 SOP (sum-of-pairs)打分系统的目标函数对比对结果进行评估。

(1) 如果没有参考比对的结果，使用下面标准的 sum-of-pairs 打分函数：

$$SOP=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}D(l_i,l_j) \tag{6.32}$$

式中，l_i 是已比对的序列；D 为距离矩阵。

为了避免在比对过程中空位的积聚，从 SOP 分数中推演出仿射几何学的空位代价，对于比对结果中一条序列的空位代价按照下面的公式进行计算：

$$Gap \quad \boldsymbol{cost} = GOP + n \times GEP \tag{6.33}$$

式中，GOP 表示第一个起始空位的固定罚分；GEP 表示对于扩展的空位的罚分；n 为一条序列中空位的个数。

对于已比对的每条序列的空位都要计算相应的空位代价。多序列比对结果的 SOP 的分值减去空位代价的总和，即为 SOP 的分值。

(2) 如果有参考比对，使用下面的 SOP 打分函数 SPS：

设有 N 个比对测试序列，构成 M 栏，标记第 i 栏的比对列为 $A_{i1}, A_{i2}, \cdots, A_{iN}$。对每一对残基 A_{ij} 和 A_{ik}，定义变量 p_{ijk}，如果 A_{ij} 和 A_{ik} 在比对的参考结果中也位于同一列，则 $p_{ijk} = 1$，否则 $p_{ijk} = 0$。定义变量 S_i 为第 i 栏的得分，则有

$$S_i = \sum_{j=1, j \neq k}^{N} \sum_{k=1}^{N} p_{ijk} \tag{6.34}$$

记 SPS 为最终比对结果的得分，则

$$SPS = \frac{\sum_{i=1}^{M} S_i}{\sum_{i=1}^{M_r} S_{ri}} \tag{6.35}$$

式中，M_r 是参考比对结果中的栏数；S_{ri} 是参考比对结果中第 i 栏的得分。

用该函数进行评价时，比对结果的得分越多，说明比对的结果越好。

图 6.6 分别给出了核酸序列 High_short 在 6 种算法 CW、BW、PSO、QPSO、DMQPSO、DGQPSO 下得到的序列比对结果；图 6.7 分别给出了蛋白质序列 451c 在 6 种算法 CW、BW、PSO、QPSO、

DMQPSO、DGQPSO 下得到的序列比对结果。从图中可以看出，DGQPSO 算法找到的相似性区域最多。

```
SEQ5            CTGGCATTCGAGTCATTTTCAGCGTTTTCGGGCCTCGATCGGCCGATGGGCGGGCGACTA
SEQ9            ----------------------------------CGGCTCGAATGCTTGGAGAAAGATCA
SEQ12           ---------------GAGTAAATATATCGTAATATAAATTGAGCGCT------AAATTCA
                                                      * *   * *            *

SEQ5            AGTATTTACAATTCTGCACCAGACCAACCA-AATGCAGCATTCACTGTGGCCAATACTGA
SEQ9            TATCTTTAACAAGCTGTACTCTACTGGGAACAAAGGAGCGCCGGACGTGTCTAATTCATG
SEQ12           GGTCCTCTCTAAATTATGGCCGGCCGCGTA---------GTTACTCTTATGTGACGGATG
                  *  *    *   *        *     *                 *     *

SEQ5            ACATAAGTAATTGTCGTGACGGAAGGCTGCGAACGTCTCAGGGATAGCAACTTCAGCGCG
SEQ9            ACCCAGATGATGCTC---------------------------------------------
SEQ12           GCAGGCTGGTACCTCGAAACGGGTACTACTGAGTGTACTCGAAATAACATGGTCAGGAGC
                 *           **

SEQ5            TTCTCTCCCCATCCAGGTTAAGAACTCAGCAGACGGGTCGTTGGAACAGGATCTACGTGG
SEQ9            ----------------------------------------------CGAGCTATGCGGAG
SEQ12           AAAGACTCTTCTTTCTTATCAACTTTCACC----------------CTGTCCCCGCGCGG
                                                              *        **  *

SEQ5            ACTAAACAACGAACAATGAGCGACGTAGCGACTCCCGTATCTCAAACTCATGGAGTTGGT
SEQ9            ATTAAATACTGAA----------------------------------TCAACGAGGTCAA
SEQ12           TTTAGGCAGC-------------------------------------CCGATGGGATAGT
                  **   *                                        *   * * *

SEQ5            GACAACGAGTTTATGCGTCTATAACGCGGTTGTGGGCTCGGTAAGCGATACGTGGATAGC
SEQ9            GAGATGGGTTTGATCCCGCTTCAGC------------------------------ATATC
SEQ12           CCCGAGGAGTTGGTTACACGTTACC----------------------GTTAATTAGGAGT
                      *  **  *    *   * *                                *
SEQ5            GTGGATCGCGACGACAGTTTGTTGTGCGCTCGAACGAACAACTCGTGATACCGGTATGCT
SEQ9            GCAGATTACATCCA-------------------------AACTTGTAACACCGGTG-ACT
SEQ12           ATTAATCCAACCCACAAATAGAAGTGCGTGTACATGG--CATTAGTTATAGGGG------
                    **     * *                          * * ** * *  **

SEQ5            CAGCTACAGACACCTGTTATCGACATGTATAAAAGGAGGTCAAAACCCAGTTCAGCCACG
SEQ9            GAGCTTAA--------------------------------CAGTATCCCGCGCATTTATG
SEQ12           ----------------------------------------GAACATTCGGCACATTGGAA
                                                         *  *  * *  **

SEQ5            GCGTAATCATTAATATG---TATTCAGATGCCGGGTCTAAAAGACATGACACTGTAGGAC
SEQ9            AC-------TTGCTAGG---CGCGCGCGTGATAGTAATAATTTACACGAGCCAGTAG---
SEQ12           CTGTAACCGTTACTTAGTCATGTCCGCGGGCCCATATCTAGATGTGGGC-----------
                         **  *  *       *    *         *       *

SEQ5            GAAAGGTAAGAAAGGCTACCTACCAATTCGATTGTGGCCGGAATACAGGGTAGAACGGCG
SEQ9            ------------------------------------------------------------
SEQ12           ------------------------------------------------------------

SEQ5            TTTAA
SEQ9            -----
SEQ12           -----
```

(a)　BW 算法下的序列比对结果

图 6.6　核酸序列 High_short 在五种算法 CW、BW、PSO、QPSO、DMQPSO、DGQPSO 下得到的序列比对结果

```
SEQ5            CTGGCATTCGAGTCATTTTCAGCGTTTTCGGGCCTCGATCGGCCGATGGGCGGGCGACTA
SEQ9            ------------------------------------------------------------
SEQ12           ------------------------------------------------------------


SEQ5            AGTATTTACAATTCTGCACCAGACCAACCAAATGCAGCATTCACTGTGGCCAATACTGAA
SEQ9            ------------------------------------------------------------
SEQ12           ------------------------------------------------------------


SEQ5            CATAAGTAATTGTCGTGACGGAAGGCTGCGAACGTCTCAGGGATAGCAACTTCAGCGCGT
SEQ9            ------------------------------------------------------------
SEQ12           -----------------------------GAGTAAATATATCGTAATATAAATTGAGCGC


SEQ5            TCTCTCCCCATCCAGGTTAAGAACTCAGCAGACGGGTCGTTGGAACAGGATCTACGTGGA
SEQ9            ---------------------------------------C--GGCTCGAATGCT-TGGA-
SEQ12           TAAATTCAGGTCCTCTCTAAATTATGGCCGGCCGCGTAGT--TACTCTTATGTGACGGAT
                                                                 **      *

SEQ5            CTAAACAACGAACAATGAGCGACGTAGCGACTCCCGTATCTCAAACTCATGGAGTTGGTG
SEQ9            -GAAAGATCATATCTTTAA-CAAGCTGTACTC----TA-CTGGGAACAAAGGAGCGCCGG
SEQ12           GGCAGGCTGGTACCTCGAAACGGGTACTACTGAGTGTA-CTCGAAATAACATGGTCAGGA
                   *       *     *     *            ** **   *   *    *

SEQ5            ACAACGAGT-TTATGCGTCTATAACGCGGTTGTGGGCTCGGTAAGCGATACGTGGATAGC
SEQ9            ACGT-GTCTAATTCATGACCCAGATGATGCTCCGAGCTATGCGGAGATTAAATA--CTGA
SEQ12           GCAAAGACTCTTCTTTCTTATCAACTTTCACCCTGTCCCCGCGCGGTTTAGGCAGCCCGA
                 *   *  *  *           *            *   *       **        *


SEQ5            GTGGATCGCGACGACAGTTTG-TTGTGCGCTCGAACGAACAACTCGTGATACCGGTATGC
SEQ9            ATCAACGAGGTCAAGAGATGGGTT---TGATCCCGCTTCAGCATATCGCAGATTACATCC
SEQ12           TGGGATAGTCCCGAGGAGTTGGTTACACGTTACCGTTAATTAGGAGTATTAATCCAACCC
                    *      * *    * * **    * *                         *  *

SEQ5            TCAGCTACAGACACCTGTTATCGACATGTATAAAAGGAGGTCAAAACCCAGTTCAGCCAC
SEQ9            A-AACTTGTAACACCGGTGACTGAGCTTAACAGTATCCCG--CGCATTTATGACTTGCTA
SEQ12           ACAAATAGAAGTGCGTGTACATGGCATTAGTTATAGGGGG--AACATTCGGCACATTGGA
                  *  *       *  **    *   *       *    *     *       *

SEQ5            GGCGTAATCATTAATATGTATTCAGATGCCGGGTCTAAAAGACATGACACTGTAGGACGA
SEQ9            GGCGCGCGTGATAG--TAATAATTTACACGAGCCAGTAG---------------------
SEQ12           ACTGTAACCGTTACT---TAGTCATGTCCGCGGGCCCATATCTAGATGTGGGC-------
                   *    * * *     ** * *  * *  *  *    *

SEQ5            AAGGTAAGAAAGGCTACCTACCAATTCGATTGTGGCCGGAATACAGGGTAGAACGGCGTT
SEQ9            ------------------------------------------------------------
SEQ12           ------------------------------------------------------------


SEQ5            TAA
SEQ9            ---
SEQ12           ---
```

(b)　CW 算法下的序列比对结果

图 6.6　核酸序列 High_short 在五种算法 CW、BW、PSO、QPSO、DMQPSO、DGQPSO 下得到的序列比对结果(续)

```
SEQ5            CTGGCATTCGAGTCATTTTCAGCGTTTTCGGGCCTCGATCGGCCGATGGGCGGGCGACTA
SEQ9            ----------------------------------CGGCTCGAATGCTTGGAGAAAGATCA
SEQ12           ---------------GAGTAAATATATCGTAATATAAATTGAGCGCT------AAATTCA
                                                      * *   * *            *

SEQ5            AGTATTTACAATTCTGCACCAGACCAACCAAATGCAGCATTCACTGTGGCCAATACTGAA
SEQ9            TATCTTTAACAAGCTGTAC--------------------TCTACTGGAACA---------
SEQ12           GGTCCTCTCTAAATTATGGCCGGCCGCGTAGTTAC----TCTTATGTGACGG--------
                  *  *    *   *                        *    ** *

SEQ5            CATAAGTAATTGTCGTGACGGAAGGCTGCGAACGTCTCAGGGATAGCAACTTCAGCGCGT
SEQ9            -----------------AAGGAGCGCCGGACGTGTCTAA---------------------
SEQ12           -----------------ATGGCAGGCTG---GTACCTCG-AAACGGGTACTACTGAGTGT
                                 * **   ** *       **

SEQ5            TCTCTCCCCATCCAGGTTAAGAACTCAGCAGACGGGTCGTTGGAACAGGATCT-------
SEQ9            ----------------TTCATGACCCAGATGATGCTCCG--------AGCTAT-------
SEQ12           ACTCGAAATAACATGGTCAGGAGC---AAAGACTCTTCTTTCTTATCAACTTTCACCCTG
                                *      *      **     *            * *

SEQ5            -----ACGTGGACTAAACAACGAACAATGAGCGACGTAGCGACTCCCGTATCTCAAACTC
SEQ9            -----GCGGAGATTAAATACTGAAT-----------------------------------
SEQ12           TCCCCGCGCGGTTTAGGCAGCC--------------------------------------
                      **  *  **   *

SEQ5            ATGGAGTTGGTGACAACGAG---TTTATGCGTCTATAACGCGGTTGTGGGCTCGGTAAGC
SEQ9            -------------CAACGAG---GTCAAGAG--------------ATGGGTTTGATCC-C
SEQ12           -------------CGATGGGATAGTCCCGAG----------------GAGTTGGTTACAC
                             * * * *    *   * *                * * * * *   *

SEQ5            GATACGTGGATAGCGTGGATCGCGACGACAGTTTGTTGTGCGCTCGAACGAACAACTCGT
SEQ9            GCTTC-AGCATATCGCAGATTACATCCA-------------------------AACTTGT
SEQ12           GTTAC--------CGTTAATTAGGAGTA-----------TTAATCCAACCCACAAATAGA
                * * *        **   **       *                         ** * *

SEQ5            GATACCGGTATGCTCAGCTACAGACACCTGTTATCGACATGTATAAAAGGAGGTCAAAAC
SEQ9            AACACCGGTGACT----------------------GAGCTTAAC-----------AGTAT
SEQ12           AGTGCGTGTACAT----------------------GGCATTAGTTATAGGGGG--AACAT
                    *  **                          *   *               *  *

SEQ5            CCAGTTCAGCCACGGCGTAATCATTAATATGTATTCAGATGCCGGGTCTAAAAGACATGA
SEQ9            CCCGCGCATTTATGAC-------TTGCTAGGCGCGCGCGTGATAGTAATAATTTACACGA
SEQ12           TCGGCACATTGGAACTGTAACCGTTACTTAG-----------------TCATGTCCGCGG
                 * *  **               **  *  *                 * *    *  *

SEQ5            CACTGTAGGACGAAAGGTAAGAAAGGCTACCTACCAATTCGATTGTGGCCGGAATACAGG
SEQ9            GCCAGTAG----------------------------------------------------
SEQ12           GCCCATA------------------------------TCTAGATGTGGGC----------
                  *  **

SEQ5            GTAGAACGGCGTTTAA
SEQ9            ----------------
SEQ12           ----------------
```

(c)　PSO 算法下的序列比对结果

图 6.6　核酸序列 High_short 在五种算法 CW、BW、PSO、QPSO、DMQPSO、DGQPSO 下得到的序列比对结果(续)

```
SEQ5           CTGGCATTCGAGTCATTTTCAGCGTTTTCGGGCCTCGATCGGCCGATGGGCGGGCGACTA
SEQ9           CGG----------------------------------CTCGAATGCTTGGAGAAAGATCA
SEQ12          GAG---------------TAAATATATCGTAATATAAATTGAGCGCT------AAATTCA
                 *                                   * *   * *            *

SEQ5           AGTATTTACAATTCTGCACCAGACCAACCA-AATGCAGCATTCACTGTGGCCAATACTGA
SEQ9           TATCTTTAACAAGCTGTACTCTACTGGGAACAAAGGAGCGCCGGACGTGTCTAATTCATG
SEQ12          GGTCCTCTCTAAATTATGGCCGGCCGCGTA---------GTTACTCTTATGTGACGGATG
                 *  *    *   *        *     *                 *     *

SEQ5           ACATAAGTAATTGTCGTG-----ACGGAAGGCTGCGAACGTCTCAGGGATAGCAACTTCA
SEQ9           ACCCAGATGATGCTCCGAGCTATGCGGAGATTAAATA------CTGAATCAACGAGGTCA
SEQ12          GCAGGCTGGTACCTCGAA-----ACGGGTACTACTGAGTGTACTCGAAATAACATGGTCA
                *           **         ***         *        *    * *    ***

SEQ5           GCGCGTTCTCTCCCCATCCAGGTTAAGAACTCAGCAGACGGGTCGTTGGAACAGGATCTA
SEQ9           -----------------------------------------------AGAGATGGGTTTG
SEQ12          -----------------------------------------------GGAGCAAAGACTC
                                                               **        *

SEQ5           CGTGGACTAAACAA--------CGAACAATGAGCGACGTAGCGACTCCCGTATCTCAAAC
SEQ9           ATCCCGCTTCAGCA----------------------------------------------
SEQ12          TTCTTTCTTATCAACTTTCACCCTGTCCCCGCGCGGTTTAGGCAGCCCGATGGGATAGTC
                     **     *

SEQ5           TCATGGAGTTGGTGACAACGAGTTTATGCGTCTATAACGCGGTTGTGGGCTCGGTAAGCG
SEQ9           ------------------------------------------------------------
SEQ12          CCGAGGAGTTGGTTACACG-----------------------------------------

SEQ5           ATACGTGGATAGCGTGGATCGCGACGACAGTTTGTTGTGCGCTCGAACGAACAACTCGTG
SEQ9           ---------TATCGCAGATTACATCCA-------------------------AACTTGTA
SEQ12          --------TTACCGTTAATTAGGAGTAT-----------TAATCCAACCCACAAATAGAA
                        ** **   **       *                         ** * *

SEQ5           ATACCGGTAT----GCTCAGCTACAGACACCTGTTATCGACATGTATAAAAGGAGGTCAA
SEQ9           ACACCGGTGACTGAGCTTAAC-------------------------------------AG
SEQ12          GTGCGTGTACATGGCATTAGTTATAG----------------------------GGGGAA
                  *  **        * *                                       *

SEQ5           AACCCAGTTCAGCCACGGCGTAATCATTAATATGTATTCAGATGCCGGGTCTAAAAGACA
SEQ9           TATCCCGCGCATTTATGAC-------TTGCTAGGCGCGCGCGTGATAGTAATAATTTACA
SEQ12          CATTCGGCACATTGGAACTGTAACCGTTACT-----------------TAGTCATGTCCG
                *  * *  **               **  *                    * *    *

SEQ5           TGACACTGTAGGACGAAAGGTAAGAAAGGCTACCTACCAATTCGATTGTGGCCGGAATAC
SEQ9           CGAGCCAGTAG-------------------------------------------------
SEQ12          CGGGCCCATATCTAGA--------------------------------------------
                *   *  **

SEQ5           AGGGTAGAACGGCGTTTAA
SEQ9           -------------------
SEQ12          ------------TGTGGGC
```

(d)　QPSO 算法下的序列比对结果

图 6.6　核酸序列 High_short 在五种算法 CW、BW、PSO、QPSO、DMQPSO、DGQPSO 下得到的序列比对结果(续)

```
SEQ5    CTGGCATTCGAGTCATTTTCAGCGTTTTCGGGCCTCGATCGGCCGATGGGCGGGCGACTA
SEQ9    C-GG--CTCGAAT-----------------------GCT-------TGG-AG------AA
SEQ12   G---------AGTAAATAT-ATCGTAATA-----TAAAT-------TGAGCG-----CTA
                  * *                         *       **   *       *

SEQ5    AGTATTTACAATTCTGCACCAGACCAACCAAAT-GCA------G----CATTC-ACTGTG
SEQ9    AG--AT--CATATCT--------TTAA-CAA---GCT------G----TACTCTACTG--
SEQ12   AA--TT--CAGGTCCTCT-----CTAAATTA-TGGCCGGCCGCGTAGTTACTCTTATG--
        *    *  **  **           **   *   **       *     * **   **

SEQ5    GCCAATACTGAACATAAGTAATTGTCGTGACGGAAGGCTG------C-GAACGTCTCAGG
SEQ9    --------GGAACA---------------AAGGAGCGCC---------GGACGTGTCT--
SEQ12   --------TGACGG---------------ATGGCAGGCTGGTACCTCGAAACGGGTAC--
                 **                  * **   **            ***  *

SEQ5    GATAGCAACTTCAGCGCGTTCTCTCCCC-ATCC-AGGTT--AAGAACTCAGCAGACGGGT
SEQ9    -------AATTCAT-------------G-ACCC-AGATG--ATG--CT------------
SEQ12   ------TACTGAGT-------------GTACTCGAAATAACATG--GT------------
               * *                   *  * *  *   * *   *

SEQ5    CGTTG-GAACAGGATCTACGTGGACTAAACAACGAACAATGAGCGACGTAGCGACTCCCG
SEQ9    ---CC-GA----GCTATGCGGAGATTAAATACTGAA---TCA------------------
SEQ12   ---CAGGA----GCAA-----------------------------------AGACTCTTC
              **    *

SEQ5    TATCTCAAACTCATGGAGTTGGTGACAACGAGTTTATGCGTCTATAACGCGGTTGTGGGC
SEQ9    ----------------------------------------------ACGAGGTC-AAGA-
SEQ12   TTTCT------TATCAACTT----TCACCCTGTCC---------CCGCGCGGTT-TAGGC
                                                       ** ***    *

SEQ5    TCGGTAAGCGATACGTGGATAGCGTGGATCGCGACGACAGTTTGTTGTGCGCTCGAACGA
SEQ9    ---------GAT--G-GGTTTG------ATCCC----------GCTTCAGCAT---ATCG
SEQ12   A--G--CCCGAT--G-GGATAG------TCCCGAGG--AGTTGGTTACACGTT---ACCG
                 ***  * ** * *         *           * *      *   *

SEQ5    AC-------------------AACTCGTGATACCGGTATGCTCAGCTACAGACACCTGTT
SEQ9    CAGATTA--------CATCCAAACTTGTAACACCGGTGA-CTGAG---------------
SEQ12   TTAATTAGGAGTATTAATCCAACCCACAAATAGAAGTGC-GTGTA---------------
                             * *     * *   **    *

SEQ5    ATCGACATGTATAAAAGGAGGTCAAAACCCAGTTCAGCCACGGCGTAATCATTAATATGT
SEQ9    --------------------------------------CTTAAC------AGTA------
SEQ12   --------------------------------------CATGGC------ATTAGT---T
                                              *    *      * **

SEQ5    ATTCAGATGCCGGGTCTAAAAGACATGACACTGTAGGACGAAAGGTAAGAAAGGCTAC-C
SEQ9    -T----C----CCG---CGCATTTATGACTTGCT------AGGCGCGC---GCGTGATAG
SEQ12   AT--AG--G----G---GGAACATTCGGCACATTGG----AACTGTAA---CCGTTAC-T
         *           *      *     * *    *      *   *        *  *

SEQ5    TACCAATTCGATTGTG-GCCGGAATACAGGGTAGAACGGCGTTTAA
SEQ9    TAATAATTT--ACACGAGCC-------A--------------GTAG
SEQ12   TAGTCATGT--CCGCGGGCCCA--T--A-TCTAGA----TGTGGGC
        **   **        * ***       *
```

(e)　DMQPSO 算法下的序列比对结果

图 6.6　核酸序列 High_short 在五种算法 CW、BW、PSO、QPSO、DMQPSO、DGQPSO 下得到的序列比对结果(续)

```
SEQ5            CTGGCATTCGAGTCATTTTCAGCGTTTTCGGGCCTCGATCGGCCGATGGGCGGGCGACTA
SEQ9            ----------------------------------CGGCTCGAATGCTTGGAGAAAGATCA
SEQ12           ---------------GAGTAAATATATCGTAATATAAATTGAGCGCT------AAATTCA
                                                      * *   * *            *

SEQ5            AGTATTTACAATTCTGCACCAGACCAACCAAATGCAGCATTCACTGTGGCCAATACTGAA
SEQ9            TATCTTTAACAAGCTGTAC--------------------TCTACTGGGAACA--------
SEQ12           GGTCCTCTCTAAATTATGGCCGGCCGCGTAGTTAC----TCTTATGTGACGG--------
                  *  *    *   *                        *    ** *

SEQ5            CATAAGTAATTGTCGTGACGGAAGGCTGCGAACGTCTCAGGGATAGCAACTTCAGCGCGT
SEQ9            -----------------AAGGAGCGCCGGACGTGTCTAA---------------------
SEQ12           -----------------ATGGCAGGCTG---GTACCTCG-AAACGGGTACTACTGAGTGT
                                 * **   ** *       **

SEQ5            TCTCTCCCCATCCAGGTTAAGAACTCAGCAGACGGGTCGTTGGAACAGGATCT-------
SEQ9            ----------------TTCATGACCCAGATGATGCTCCG--------AGCTAT-------
SEQ12           ACTCGAAATAACATGGTCAGGAGC---AAAGACTCTTCTTTCTTATCAACTTTCACCCTG
                                *      *      **     *            * *

SEQ5            -----ACGTGGACTAAACAACGAACAATGAGCGACGTAGCGACTCCCGTATCTCAAACTC
SEQ9            -----GCGGAGATTAAATACTGAAT-----------------------------------
SEQ12           TCCCCGCGCGGTTTAGGCAGCC--------------------------------------
                      **  *  **   *

SEQ5            ATGGAGTTGGTGACAACGAG---TTTATGCGTCTATAACGCGGTTGTGGCTCGGTAAGC
SEQ9            -------------CAACGAG---GTCAAGAG--------------ATGGGTTTGATCC-C
SEQ12           -------------CGATGGGATAGTCCCGAG----------------GAGTTGGTTACAC
                             * * * *    *   * *                * * * * *   *

SEQ5            GATACGTGGATAGCGTGGATCGCGACGACAGTTTGTTGTGCGCTCGAACGAACAACTCGT
SEQ9            GCTTC-AGCATATCGCAGATTACATCCA-------------------------AACTTGT
SEQ12           GTTAC--------CGTTAATTAGGAGTA-----------TTAATCCAACCCACAAATAGA
                * * *        **   **       *                         ** * *

SEQ5            GATACCGGTATGCTCAGCTACAGACACCTGTTATCGACATGTATAAAAGGAGGTCAAAAC
SEQ9            AACACCGGTGACT----------------------GAGCTTAAC-----------AGTAT
SEQ12           AGTGCGTGTACAT----------------------GGCATTAGTTATAGGGGG--AACAT
                    *  **                          *   *               *  *

SEQ5            CCAGTTCAGCCACGGCGTAATCATTAATATGTATTCAGATGCCGGGTCTAAAAGACATGA
SEQ9            CCCGCGCATTTATGAC-------TTGCTAGGCGCGCGCGTGATAGTAATAATTTACACGA
SEQ12           TCGGCACATTGGAACTGTAACCGTTACTTAG-----------------TCATGTCCGCGG
                 * *  **               **  *  *                 * *    *  *

SEQ5            CACTGTAGGACGAAAGGTAAGAAAGGCTACCTACCAATTCGATTGTGGCCGGAATACAGG
SEQ9            GCCAGTAG----------------------------------------------------
SEQ12           GCCCATA------------------------------TCTAGATGTGGGC----------
                  *  **

SEQ5            GTAGAACGGCGTTTAA
SEQ9            ----------------
SEQ12           ----------------
```

(f)　DGQPSO 算法下的序列比对结果

图 6.6　核酸序列 High_short 在六种算法 CW、BW、PSO、QPSO、DMQPSO、DGQPSO 下得到的序列比对结果(续)

```
cy2_rhoge       ---ATPAELATKAG-CAVCHQPTAKGL-GPSYQEIAKKYKGQAGAPALMAER-VRKGSV-
c550_bacsu      ---ASPEEI-YKAN-CIACHGENYEGVSGPSLKGV-----GDKKDVAEIKTK-IEKGGN-
1dvh            ---ADGAAL-YKS--CIGCHGADGSKAAMGSAKPV----KGQGAEELYKKMKGYADGSY-
1etp            DAEAGQGKV---AV-CGACHGVDGNSP-APNFPKL----AGQGERYLLKQLQDIKAGSTP
1cch            ---QDGEAL-FKSKPCAACHSVDTKMV-GPALKEVAAKNAGVEGAADTLALH-IKNGSQ-
                        :   :  *  **    .         :     *          :    *.

cy2_rhoge       ----GIFGKLPMTPTPARPISDADLKLVIDWIL---
c550_bacsu      ----G----MPSGLVPADKLDD-----MAEWVSKI-
1dvh            ----G-GERKAMMTNAVKKYSDEELKALADYMSKL-
1etp            GAPEGVGRKVLEMTGMLDPLSDQDLEDIAAYFSSQK
1cch            ----GVWGPIPM---PPNPVTEEEAKILAEWVLSLK
                    *                :     :  :.
```

(a)　BW 算法下的序列比对结果

```
cy2_rhoge       --ATPAELATKAGCAVCHQPTAKGLGPSYQEIAKKYKGQAGAPALMAERVRKGSVGIFGK
c550_bacsu      --ASPEEIYKAN-CIACHGEN--YEGVSGPS--LKGVGDKKDVAEIKTKIEKGGNGMPSG
1dvh            --ADGAALYKS--CIGCHGADGSKAAMGSAKP-VKGQGAEELYKKMKGYADGSYGGERKA
1etp            DAEAGQGKVAV--CGACHGVDGNSPAPNFPK--LAGQGERYLLKQLQDIKAGSTPGAPEG
1cch            --QDGEALFKSKPCAACHSVDTKMVGPALKEVAAKNAGVEGAADTLALHIKNGSQGVWGP
                             *  **       .    .      *       :      .  *

cy2_rhoge       LP----MTPTPARPISDADLKLVIDWIL---
c550_bacsu      L-------------VPADKLDDMAEWVSKI-
1dvh            M------MTNAVKKYSDEELKALADYMSKL-
1etp            VGRKVLEMTGMLDPLSDQDLEDIAAYFSSQK
1cch            IP-------MPPNPVTEEEAKILAEWVLSLK
                :              .  . . :  :.
```

(b)　CW 算法下的序列比对结果

```
cy2_rhoge       ATPAELATK-AG-CAVCHQP--TAKGLGP-SYQEIAKKYKGQAGAPALMA--ERVRKGS-
c550_bacsu      ASPEEIY-K-AN-CIACHGENYEGVS-GP-SLKGVGDKKDV-----AEIK--TKIEKGG-
1dvh            ADGAALY-K-S--CIGCHGA--DGSKAAMGSAKPVKGQGAE-----ELYKKMKGYADGS-
1etp            -DAEAGQGKVAV-CGACHGV--DGNSPAP-NFPKLAGQGER-----YLLKQLQDIKAGST
1cch            QDGEALF-K-SKPCAACHSV--DTKMVGP-ALKEVAAKNAGVEGAADTLA--LHIKNGS-
                        * :  *  **         .      :  :                   *.

cy2_rhoge       ----VGIFGKLPMTPTPARPISDADLKLVIDWIL---
c550_bacsu      ----NG------M---PSGLVPADKLDDMAEWVSKI-
1dvh            ----YG-GERKAMMTNAVKKYSDEELKALADYMSKL-
1etp            PGAPEGVGRKVLEMTGMLDPLSDQDLEDIAAYFSSQK
1cch            ----QGVWGPIPM---PPNPVTEEEAKILAEWVLSLK
                     *               .  . . :  :.
```

(c)　PSO 算法下的序列比对结果

图 6.7　蛋白质序列 451c 在五种算法 CW、BW、PSO、QPSO、DMQPSO、DGQPSO 下得到的序列比对结果

```
cy2_rhoge       ATPAELATKAGCAVCHQPTAKGL-GPSYQEIAKKYKGQAGAPALMAERVRKGSVGIFGKL
c550_bacsu      ASPEEIYK-ANCIACHGENYEGVSGPSLKGVGDKKD-----VAEIKTKIEKGGNG----M
1dvh            ADGAALYK--SCIGCHGADGSKAAMGSAKPVKGQGA------EELYKKMKGYADGSYGGE
1etp            DAEAGQGKVAVCGACHGVDGNSP-APNFPKLAGQGE------RYLLKQLQDIKAGSTPGA
1cch            QDGEALFKSKPCAACHSVDTKMV-GPALKEVAAKNAGVEGAADTLALHIKNGSQGVWGPI
                       .   *  **    .         :  :          :  ::.    *

cy2_rhoge       PMTP--------TPARPISDADLKLVIDWIL---
c550_bacsu      PSGL-------------VPADKLDDMAEWVSKI-
1dvh            RKAM------MTNAVKKYSDEELKALADYMSKL-
1etp            PEGVGRKVLEMTGMLDPLSDQDLEDIAAYFSSQK
1cch            PM-----------PPNPVTEEEAKILAEWVLSLK
                                  .  . . :  :.
```

(d)　QPSO 算法下的序列比对结果

```
cy2_rhoge       ATPAELATKAGCAVCHQPTAKGL-GPSYQEIAKKYKGQAGAPALMAERVR---K-GS-VG
c550_bacsu      ASPEEIYK-ANCIACHGENYEGVSGPSLKGVGDKK----DV-AEIKTKIE---K-GG-NG
1dvh            ADGAALYK--SCIGCHGADGSKAAMGSAKPVKGQG-----A-EELYKKMKGYAD-GS-YG
1etp            DAEAGQGKVAVCGACHGVDGNSP-APNFPKLAGQG--ERYLLKQLQ-DIKAGSTPGAPEG
1cch            QDGEALFKSKPCAACHSVDTKMV-GPALKEVAAKNAGVEGAADTLALHIKNGSQ-----G
                       .   *  **    .         :  :          :   :.         *

cy2_rhoge       IFGKLPMTPTPARPISDADLKLVIDWI---L
c550_bacsu      M------PSG---LVPADKLDDMAEWV-SKI
1dvh            GERKAM-MTNAVKKYSDEELKALADYM-SKL
1etp            VGRKVLEMTGMLDPLSDQDLEDIAAYFSSQK
1cch            VWGPIP-MP--PNPVTEEEAKILAEWVLSLK
                        .      .  . . :  :.
```

(e)　DMQPSO 算法下的序列比对结果

```
cy2_rhoge       ATPAELATKAGCAVCHQPTAKGL-GPSYQEIAKKYKGQAGAPALMAERVRKGSVGIFGKL
c550_bacsu      ASPEEIYK-ANCIACHGENYEGVSGPSLKGVGDKKD-----VAEIKTKIEKGGNG----M
1dvh            ADGAALYK--SCIGCHGADGSKAAMGSAKPVKGQGA------EELYKKMKGYADGSYGGE
1etp            DAEAGQGKVAVCGACHGVDGNSP-APNFPKLAGQGE------RYLLKQLQDIKAGSTPGA
1cch            QDGEALFKSKPCAACHSVDTKMV-GPALKEVAAKNAGVEGAADTLALHIKNGSQGVWGPI
                       .   *  **    .         :  :          :  ::.    *

cy2_rhoge       PMTP--------TPARPISDADLKLVIDWIL---
c550_bacsu      PSGL-------------VPADKLDDMAEWVSKI-
1dvh            RKAM------MTNAVKKYSDEELKALADYMSKL-
1etp            PEGVGRKVLEMTGMLDPLSDQDLEDIAAYFSSQK
1cch            PM-----------PPNPVTEEEAKILAEWVLSLK
                                  .  . . :  :.
```

(f)　DGQPSO 算法下的序列比对结果

图 6.7　蛋白质序列 451c 在 6 种算法 CW、BW、PSO、QPSO、DMQPSO、DGQPSO 下得到的序列比对结果(续)

6.4　本章小结

本章提出了使用 QPSO 算法及其改进的 QPSO 算法 DMQPSO 和 DGQPSO 来训练剖面隐马尔可夫模型(HMM)的参数，进行多序列的比对。HMM 算法是基于概率模型的，它本身的结构决定了它可以用在多序列比对上。从实验结果可以看出，在训练的过程中，BW 算法和 PSO 算法容易陷入局部最优，但是 QPSO 算法、DMQPSO 和 DGQPSO 算法是一种全局收敛算法，可以得到最优的模型参数，是一种非常有效的 HMM 训练方法。在比对的过程中，QPSO 算法、DMQPSO 和 DGQPSO 算法，不论 SOP 得分还是 SPS 得分，分值都高于 BW 和 PSO 算法，并且能产生较好的比对结果。从所有实验结果中还可以得出 DGQPSO 算法的性能最好，能够找到最多的相似性区域。

在训练 HMM 的时间上，PSO 算法、QPSO 算法、DMQPSO 和 DGQPSO 算法所消耗的时间大致相同，平均时间为 6h，但是这四种算法的时间都大于 BW 所消耗的时间，BW 的训练时间平均只需要几分钟。需要强调的是，基于 HMM 和 QPSO 算法、DMQPSO、DGQPSO 算法在对于长序列的问题上也得到了较好的比对性能。但是，随着序列个数和序列长度的增加，算法所消耗的时间越来越长，怎样提高算法的效率是将来工作的一个重点。

参 考 文 献

[1] Löytynoja A, Milinkovitch M C.A hidden markov model for progressive multiple alignment[J]. Bioinformatics, 2003(19): 1505-1513.

[2] Smith L, Yeganova L, Wilbur W J. Hidden Markov models and optimized sequence alignments, Comput[J]. Biol and Chem, 2003 (27):

77-84.

[3] Sun J, Xu W B, Fang W. A Diversity-Guided Quantum-Behaved Particle Swarm Optimization Algorithm[C]. In the proceeding of 6th International Conference Simulated Evolution and Learning. SEAL, 2006: 497-504.

[4] Rabiner L R. A tutorial on hidden Markov models and elected applications in speech recognition[J]. Proc of the IEEE, 1989(77): 257-286.

第 7 章　多序列比对的并行计算

随着生物信息技术的发展，长基因组生物序列数据越来越多，原有的多序列比对方法对于长序列比对的效率较差，对于超长序列或超多序列的比对问题成为当前生物信息学急需解决的问题。随着计算机科学和计算机硬件的发展提升，上面这个问题的研究也进入了一个新的阶段——并行计算。有研究表明，序列比对的多数算法都具有良好的内在并行特性，通常的思路模式是：先将整个任务分解成大量的相互独立的子任务，通过将子任务向进程的适当映射后，可使各子任务间仅仅需要少量的消息传递而彼此配合，最终再通过将所有子任务执行的结果装配成整个任务的结果。本章将根据以上思路提出几个多序列比对的并行算法。

7.1　长序列首尾分段并行比对算法

7.1.1　引言

无论什么比对算法，在比对较短序列时都可以获得较好的比对质量，但对于长序列的比对效果并不理想。在第三代测序技术以及基因组拼接技术的不断发展下，生物信息领域获得了越来越多的长基因组序列数据，长序列比对成为急需解决的问题。目前国内外对于长序列比对方法的研究还处于探索阶段，现有的方法很少，基本分为两大类。第一类是以“种子扩展法”(seed and extend)为基本思想的算法，如较早的有基于 BWT(burrows-wheeler transform)索引技

术的 BWT-SW、BWA-SW，近期的依据哈希表(hash table)的 YAHA、SAP 等。这些工具在结构变异断点的检测敏感性和准确性方面各有特点，比对效率方面也不尽相同，但都有需要改进的地方。第二类是以“分而治之”(divide-and-conquer)为基本思想的算法，将长序列分段成短序列分别比对再整合，拼接成最后的比对结果。

传统的算法对内存空间的庞大需求以及漫长的运行时间已经无法满足对这种大规模数据的处理，因此长序列比对的并行计算成为研究的一个热点问题。张法和刘志勇等提出基于 Smith-Waterman 算法的并行生物序列比对算法(PSW-DC 算法)；陈娟基于蚁群算法结合遗传算法进行并行比对计算；Silva 等提出并行小生境多目标遗传算法进行多序列比对；Blazewicz 等基于 T-Coffee 算法提出多 GPU 支持的并行比对 G-MSA 法。在并行计算中常用到“分而治之”策略，但是基于“分而治之”策略的比对结果依赖于分段点的定位，分段点并不是简单的等分，当某次分段点定位不当，将会直接影响到下一次的分段及比对，造成连锁不良效应(即前后制约效应)，由此影响到最终的比对结果。传统的做法是以贪婪算法寻找分段点，计算效率很低，于是有研究者致力于快速寻找最优分段点，龚贺华提出基于最长相似片段分割计算的 LSS-DCA 法；陈娟应用遗传算法计算序列的分段点；业宁等同时考虑横向切片和纵向分割的折中，设计了 DCA-ClustalW 算法。但是这些方法的算法构造较为复杂，不容易编程实现。对于长序列比对，构建一个既能快速定位最优分段点又结构简单的并行算法也是本章的一项研究内容。

随着测序技术的发展，有越来越多的长序列需要分析比对，只有采用大规模并行计算才能满足长序列比对要求，“分而治之”策略是常用的并行思路。基于“分而治之”策略，结合并行计算，构造长序列首尾分段并行比对算法。将长序列首尾随机分成短序列段，再分配给相应的处理器并行比对，最后拼接整合成比对结果。这种首尾分段的方式可以降低前后制约效应，随机分段的方式可以降低寻找最优分段点的难度。

7.1.2　构造原理

基于“分而治之”策略，结合并行化思路，提出长序列首尾分段并行比对的设想，先将长序列首尾随机分成短序列段，再分配给相应的处理器并行比对，最后拼接整合成比对结果。构造理念是：分段顺序以首尾两端同时向中间逐步分段，则“首”的分段不影响到“尾”的分段，由此可降低分段的前后制约效应；先随机分段再并行计算，从中挑选最优比对以确定分段点的方式可以降低定位分段点的编程难度。具体描述如下：按照当今流行的短序列比对软件的序列长度默认推荐值 100bp 来确定基本分段长度，将长序列首尾随机分成 100bp 左右的段(在 80～120bp 之间产生随机整数作为每一段的长度，取值范围可以根据序列信息适当调整)，将这些随机分段的短序列分配到处理器并行比对，从中挑选最优比对结果，由此确定首尾分段点的定位，再按照这个定位对中间未比对的部分继续首尾分段比对，依次类推，直至全部分段比对完成，再将所有的短序列比对结果拼接形成最终的比对结果。示意图见图 7.1。

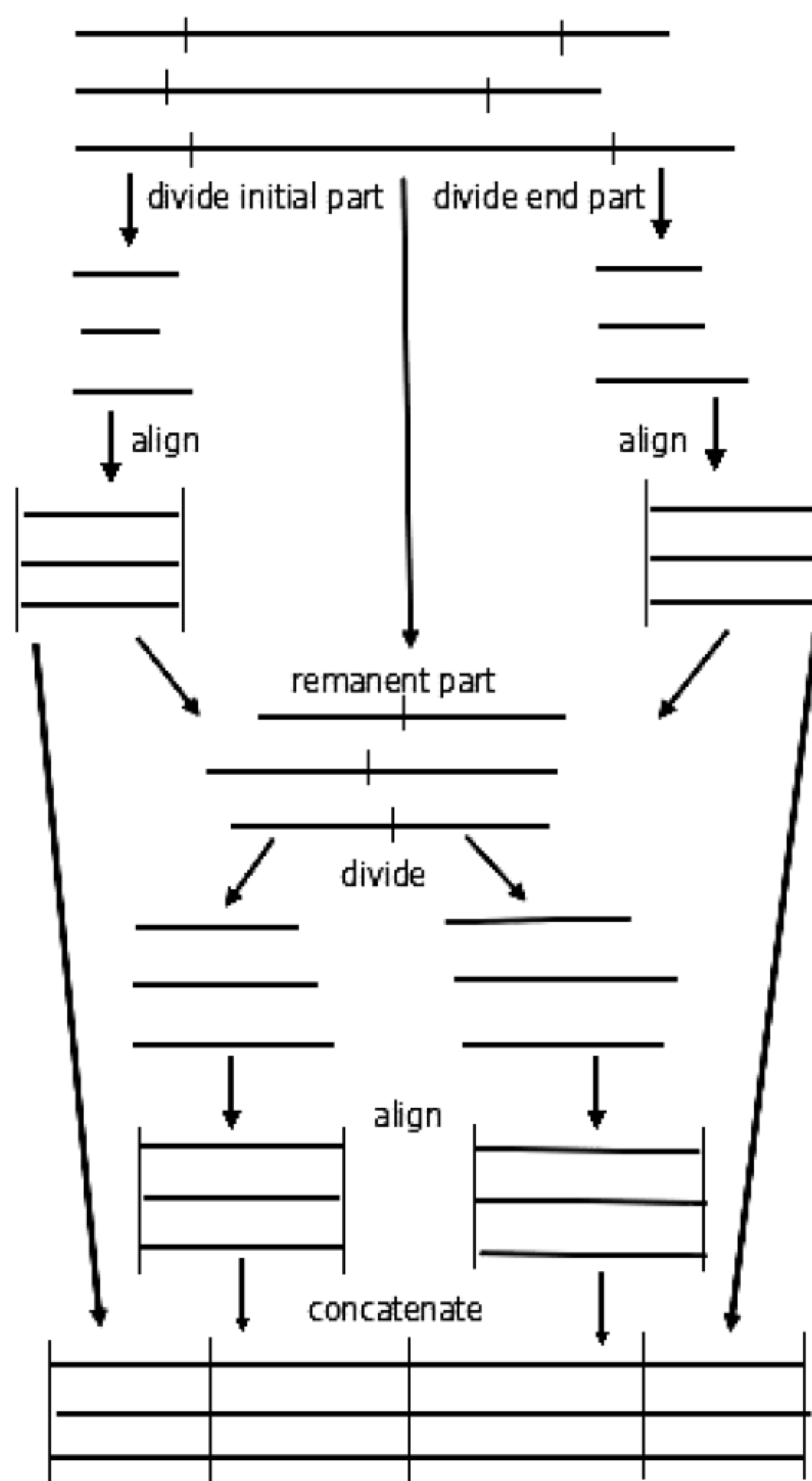


图 7.1　长序列分段比对图

7.1.3 数值模拟结果

以序列 2cba_ref1 为例，它的序列中最长序列长度为 259bp，符合长序列的特点，如图 7.2 所示。

```
CAH1_HUMAN    DWGYDDKNGPEQWSKLYPIANGNNQSPVDIKTSETKHDTSLKPISVSYNPATAKEIINVGHSFHVNFEDNDNRSVLKGGPFSDSYRLFQF
CAH4_RAT      HWCYEIQAKEPNSHCSGPEQWTGDCKKNQQSPINIVTSKTKLNPSLTPFTFVGYDQKKKWEVKNNQHSVEMSLGEDIYIFGGDLPTQYKA
CAH6_HUMAN    DWTYSEGALDEAHWPQHYPACGGQRQSPINLQRTKVRYNPSLKGLNMTGYETQAGEFPMVNNGHTVQIGLPSTMRMTVADGIVYIAQQMH
CAH_DUNSA     HDYNYEKVGFDWTGGVCVNTGTSKQSPINIETDSLAEESERLGTADDTSRLALKGLLSSSYQLTSEVAINLEQDMQFSFNAPDEDLPQLT
CAH2_CHLRE    HSLNGENWEGKDGAGNPWVCKTGRKQSPINVPQYHVLDGKGSKIATGLQTQWSYPDLMSNGSSVQVINNGHTIQVQWTYDYAGHATIAIP

CAH1_HUMAN    HFHWGSTNEHGSEHTVDGVKYSAELHVAHWNSAKYSSLAEAASKADGLAVIGVLMKVGEANPKLQKVLDALQAIKTKGKRAPFTNFDPST
CAH4_RAT      IQLHLHWSEESNKGSEHSIDGKHFAMEMHVVHKKMTTGDKVQDSDSKDKIAVLAFMVEVGNEVNEGFQPLVEALSRLSKPFTNSTVSESC
CAH6_HUMAN    FHWGGASSEISGSEHTVDGIRHVIEIHIVHYNSKYKTYDIAQDAPDGLAVLAAFVEVKNYPENTYYSNFISHLANIKYPGQRTTLTGLDV
CAH_DUNSA     IGGVVHTFKPVQIHFHHFASEHAIDGQLYPLEAHMVMASQNDGSDQLAVIGIMYKYGEEDPFLKRLQETAQSNGEAGDKNVELNSFSINV
CAH2_CHLRE    AMRNQSNRIVDVLEMRPNDASDRVTAVPTQFHFHSTSEHLLAGKIFPLELHIVHKVTDKLEACKGGCFSVTGILFQLDNGPDNELLEPIF

CAH1_HUMAN    LLPSSLDFWTYPGSLTHPPLYESVTWIICKESISVSSEQLAQFRSLLSNVEGDNAVP
CAH4_RAT      LQDMLPEKKKLSAYFRYQGSLTTPGCDETVIWTVFEEPIKIHKDQFLEFSKKLYYDQEQKLN
CAH6_HUMAN    QDMLPRNLQHYYTYHGSLTTPPCTENVHWFVLADFVKLSRTQVWKLENSLLDHRNKT
CAH_DUNSA     ARDLLPESDLTYYGYDGSLTTPGCDERVKWHVFKEARTVSVAQLKVFSEVTLAAHPEAT
CAH2_CHLRE    ANMPTREGTFTNLPAGTTIKLGELLPSDRDYVTYEGSLTTPPCSEGLLWHVMTQPQRISFGQWNRYRLAVGEKECNSTE
```

图 7.2 蛋白质序列 2cba_ref1

以每一段 100bp 左右的长度，将 2cba_ref1 随机分为三段，分别比对，结果如图 7.3 所示。

```
CAH1_HUMAN    _DWGYDDK________NGPEQWSKLYPIANGN_NQSPVDIKTSE____________TKHDTSLKPISVSYN__PATAKEIINVGHSFHVNF
CAH4_RAT      _HWCYEIQAKEPNSHCSGPEQWTGD___CKKN_QQSPINIVTSK____________TKLNPSLTPFTFVGYD__QKKKWEVKNNQHS____
CAH6_HUMAN    _DWTYSEGA_______LDEAHWPQHYPACGGQ_RQSPINLQRTK____________VRYNPSLKGLNMTGYET_QAGEFPMVNN_______
CAH_DUNSA     HDYNYEK__________VGFDWTGGVCVNTGTSKQSPINIETDSLAEESERLGTADDTSRLALKGLLSSSY___QLTSEVAINL_______
CAH2_CHLRE    HSLNGENW________EGKDGAGNPWVCKTGR_KQSPINVPQYH______VLDGKGSKIATGLQTQWSYPDLMSNGSSVQVINNGH_____
```

(a) 序列 2cba_ref1 第一段比对结果

```
CAH1_HUMAN    ________EDNDNR_____________________SVLKGGPFSDSYRLFQFHFHWG__STNEHGSEHTVDGVKYSAELHVAHWNSAKYS
CAH4_RAT      ___VEMSLGED________________________IYIFGGDLPTQYKAIQLHLHWS__EESNKGSEHSIDGKHFAMEMHVVHKKMTTGD
CAH6_HUMAN    GHTVQIGLPSTMRM_________________________TVADGIVYIAQQMHFHWGGASSEISGSEHTVDGIRHVIEIHIVHYNSKYKT
CAH_DUNSA     EQDMQFSFNAPDEDL____________________PQLTIGGVVHTFKPVQIHFHH_______FASEHAIDGQLYPLEAHMVMASQNDGS
CAH2_CHLRE    __TIQVQWTYDYAGHATIAIPAMRNQSNRIVDVLEMRPNDASDRVTAVPTQFHFH________STSEHLLAGKIFPLELHIVHKVTDKLE

CAH1_HUMAN    SLAEAASKADGLAVIGVLMKVG
CAH4_RAT      KVQDSDSK_DKI__________
CAH6_HUMAN    YDIAQDAP_DGL__________
CAH_DUNSA     _________DQLAVIGIMYKYG
CAH2_CHLRE    ACKG_____GCFSVTGILFQL_
```

(b) 序列 2cba_ref1 第二段比对结果

图 7.3 蛋白质序列 2cba_ref1 分三段分别比对

```
CAH1_HUMAN    __________E__ANPKLQKVLDALQAIKTKGKRA_____PFTNFDPSTLLPSS___LDFWTYPGSLTHPPLYESVTWIICKESISVSSE
CAH4_RAT      AVLAFMVEVGN_EVNEGFQPLVEALSRLSKPFTNS_____TVSESCLQDMLPEKKKLSAYFRYQGSLTTPGCDETVIWTVFEEPIKIHKD
CAH6_HUMAN    AVLAAFVEVKNYPENTYYSNFISHLANIKYPGQRT_____TLTGLDVQDMLPRN__LQHYYTYHGSLTTPPCTENVHWFVLADFVKLSRT
CAH_DUNSA     __________E__EDPFLKRLQETAQSNGEAGDKN_VELNSFSINVARDLLPES__DLTYYGYDGSLTTPGCDERVKWHVFKEARTVSVA
CAH2_CHLRE    _____________DNGPDNELLEPIFANMPTREGTFTNLPAGTTIKLGELLPSD___RDYVTYEGSLTTPPCSEGLLWHVMTQPQRISFG

CAH1_HUMAN    QLAQFRSLLSNVEGDNAVP
CAH4_RAT      QFLEFSKKLY_YDQEQKLN
CAH6_HUMAN    QVWKLENSLLDHRNKT___
CAH_DUNSA     QLKVFSEVTLAAHPEAT__
CAH2_CHLRE    QWNRYRLAVGEKECNSTE_
```

(c) 序列 2cba_ref1 第三段比对结果

图 7.3　蛋白质序列 2cba_ref1 分三段分别比对(续)

经过遗传算法不断迭代比对，最后将三段比对结果按顺序拼接在一起，去除同空位列，比对结果如图 7.4 所示。

```
CAH1_HUMAN    __DWGYDDKNGPEQ__WS__KLYPIANGN__NQSPVDI__KTSETKHDTSLKPISVSYNPATAK__EIINVG__HSFHVNFEDNDNRSVL
CAH4_RAT      HW_CYEIQAKEPNSHCSGPEQWTGDCKKNQQS__PINIVTSKTKLNPSLTPFTFVGYDQKKKWEVKNNQHSVE__MSLGED______IYI
CAH6_HUMAN    DW__TY_SEGALDEAHWPQHY__PAC__GGQRQSPINLQR__TKVRYNPSLKGLNMTGYETQAGEFPMVNNGHTVQIGLPS______T__
CAH_DUNSA     H__DYNYEKVGFDWTGG__V________CVNTGTSKQSPINIETDSLAEESERLGTA__DDTSRLALKGLLSSSYQLTSEV______AIN
CAH2_CHLRE    HSLNGE_NWEGKDGAGNP__WVC__KTG__RKQSPINVPQYHVLDGKGSKIATGLQTQW__SYPDLMSNGSSVQVINNGHT______IQV

CAH1_HUMAN    KGG____PFSDSY____RLFQFHFHWGSTNEHGSEHTV____DGVKYSAELH____VAHW_NSAKYSS_LAEAASK____ADGLAVIGVL
CAH4_RAT      FGGDLPTQYKAIQ____LHL____HWSEESNKGSEHSIDGKH____FA____ME____MHVVHK_KMTTGDKVQDSDSKDKIAVLAFMVE
CAH6_HUMAN    ___________MRMTVA_DGIVYI_AQQ_MHFH__WGGASSEISG_S_EHTVDGIRHVIEIHIV_HYNS_KYKTYDI_____________
CAH_DUNSA     L__EQDMQFSFNA__PDEDLPQLTIGGVVHTFKPVQI__H__FHHFASEHAIDGQL__YP__LEAHMVMASQNDGSDQLAVIGIMYKYGE
CAH2_CHLRE    QWTYDYAGHATIAIPAMRNQSNRIVDVLEMRPNDASDRVTAVPTQFHFHSTSEHLLAGKIFPLELHIV__HKVTDKLEACKGGCFSVTG_

CAH1_HUMAN    MKVGEANPKLQKV____LDALQAIKTKGKRAPFTNFDPS_____TLLPSSLDFWTYPGS_LTH____PPLYES____VTWIICKESISVS
CAH4_RAT      VGNEVNEGFQPLVEAL____SRLSKPFTNSTVSESCLQDML____PEKKKLSAY____F____RYQGSLTTPGCDETV____IWTVFEEP
CAH6_HUMAN    _AQDAP_DGL_AVLAAFVEVKNYPENTYYSNFISHLANIKYPGQRTTLTGLDVQD__MLPRNLQHYYTYHGSLTTPPCTEN__VHWFVL_
CAH_DUNSA     __EDPFLKRLQETAQSNGE____AGDKNVELNSFSINVA____R____DLLPESDL____TYYGYD____GSLTTPGCDERVKWH_VFKE
CAH2_CHLRE    ___________ILFQLDNG__PDNELLEPIFANMP__TREGTFTNLPAGTTIKLGELL__PSDRDYVTYEGSLTTPPCSEGLLWHVMTQ_

CAH1_HUMAN    SE____QLAQFRSLLSNVEGDNAVP
CAH4_RAT      IKIHKDQFLEFSKKLYYD_QEQKLN
CAH6_HUMAN    ADFVKLSRTQVWKLENSLLDHRNKT
CAH_DUNSA     ARTVSVAQLKV_FSEVTLAAHPEAT
CAH2_CHLRE    PQRISFGQWNRYRLAVGEKECNSTE
```

图 7.4　蛋白质序列 2cba_ref1 三段比对结果拼接

参考库中比对好的序列如图 7.5 所示。

```
CAH1_HUMAN    _DWGYDDK________NGPEQWSKLYPIANGN_NQSPVDIKTSE____________TKHDTSLKPISVSYN___PATAKEIINVGHSFHV
CAH4_RAT      _HWCYEIQAKEPNSHCSGPEQWTGD___CKKN_QQSPINIVTSK____________TKLNPSLTPFTFVGYD__QKKKWEVKNNQHSVEM
CAH6_HUMAN    _DWTYSEGA_______LDEAHWPQHYPACGGQ_RQSPINLQRTK____________VRYNPSLKGLNMTGYET_QAGEFPMVNNGHTVQI
CAH_DUNSA     HDYNYEK__________VGFDWTGGVCVNTGTSKQSPINIETDSLAEESERLGTADDTSRLALKGLLSSSY___QLTSEVAINLEQDMQF
CAH2_CHLRE    HSLNGENW________EGKDGAGNPWVCKTGR_KQSPINVPQYH______VLDGKGSKIATGLQTQWSYPDLMSNGSSVQVINNGHTIQV

CAH1_HUMAN    NFEDNDNR_____________________SVLKGGPFSDSYRLFQFHFHWG__STNEHGSEHTVDGVKYSAELHVAHWNSAKYSSLAEAA
CAH4_RAT      SLGED________________________IYIFGGDLPTQYKAIQLHLHWS__EESNKGSEHSIDGKHFAMEMHVVHKKMTTGDKVQDSD
CAH6_HUMAN    GLPSTMRM_________________________TVADGIVYIAQQMHFHWGGASSEISGSEHTVDGIRHVIEIHIVHYNSKYKTYDIAQD
CAH_DUNSA     SFNAPDEDL____________________PQLTIGGVVHTFKPVQIHFHH_______FASEHAIDGQLYPLEAHMVMASQNDGS______
CAH2_CHLRE    QWTYDYAGHATIAIPAMRNQSNRIVDVLEMRPNDASDRVTAVPTQFHFH________STSEHLLAGKIFPLELHIVHKVTDKLEACKG__

CAH1_HUMAN    SKADGLAVIGVLMKVGE__ANPKLQKVLDALQAIKTKGKRA_____PFTNFDPSTLLPSS___LDFWTYPGSLTHPPLYESVTWIICKES
CAH4_RAT      SK_DKIAVLAFMVEVGN_EVNEGFQPLVEALSRLSKPFTNS_____TVSESCLQDMLPEKKKLSAYFRYQGSLTTPGCDETVIWTVFEEP
CAH6_HUMAN    AP_DGLAVLAAFVEVKNYPENTYYSNFISHLANIKYPGQRT_____TLTGLDVQDMLPRN__LQHYYTYHGSLTTPPCTENVHWFVLADF
CAH_DUNSA     ___DQLAVIGIMYKYGE__EDPFLKRLQETAQSNGEAGDKN_VELNSFSINVARDLLPES__DLTYYGYDGSLTTPGCDERVKWHVFKEA
CAH2_CHLRE    ___GCFSVTGILFQL____DNGPDNELLEPIFANMPTREGTFTNLPAGTTIKLGELLPSD___RDYVTYEGSLTTPPCSEGLLWHVMTQP

CAH1_HUMAN    ISVSSEQLAQFRSLLSNVEGDNAVP
CAH4_RAT      IKIHKDQFLEFSKKLY_YDQEQKLN
CAH6_HUMAN    VKLSRTQVWKLENSLLDHRNKT___
CAH_DUNSA     RTVSVAQLKVFSEVTLAAHPEAT__
CAH2_CHLRE    QRISFGQWNRYRLAVGEKECNSTE_
```

图 7.5　蛋白质序列 2cba_ref1 参考比对结果

7.1.4　结论

基于“分而治之”的分段思想，对长序列首尾随机分段，因为分段顺序是首尾两端向中间逐步分段，可以降低前后制约效应，同时也降低了分段的编程难度。将众多随机分段的短序列分配到各处理器并行比对，再从中找出最优比对，不但确定了最优分段点，同时也得到了该分段的最优比对。这个算法构造简单，易实现编程，同时充分利用短序列高效比对的优势。

7.2　本章小结

根据当前生物信息学多序列比对的超多超长等特点，结合目前的信息技术，提出并行计算的比对思路，为超长序列与超多序列的比对提供新的思路和更优的比对结果。

参 考 文 献

[1] 王勇献，王正华. 生物信息学导论[M]. 北京：清华大学出版社，2011.

[2] Lam T W, Sung W K, Tam S L, et al. Compressed indexing and local alignment of DNA[J]. Bioinformatics, 2008, 24(6): 791-797.

[3] Li H, Durbin R. Fast and accurate long-read alignment with Burrows–Wheeler transform[J]. Bioinformatics, 2010, 26(5): 589-595.

[4] Faust G G, Hall I M. YAHA: fast and flexible long-read alignment with optimal breakpoint detection[J]. Bioinformatics, 2012, 28(19): 2417-2424.

[5] 高岩. 面向长基因组序列片段的快速比对算法研究[D]. 哈尔滨工业大学，2014.

[6] Stoye J. Multiple sequence alignment with the divide-and-conquer method[J]. Gene, 1998, 211(2): GC45-GC56.

[7] Yue F, Tang J. A Divide-and-Conquer Implementation of Three Sequence Alignment and Ancestor Inference[J]. Bioinformatics and Biomedicine, Bibm, IEEE International Conference on, 2007: 143-150.

[8] Stoye J, Perrey S W, Dress A W M. Improving the divide-and-conquer approach to sum-of-pairs multiple sequence alignment[J]. Applied Mathematics Letters, 1997, 10(2): 67-73.

[9] 张法，乔香珍，刘志勇. 基于 Smith-Waterman 算法的并行分而治之生物序列比对算法[J]. 中国科学：技术科学，2004(34)：190-199.

[10] 陈娟，陈崚. 渐进方法结合蚁群算法求解多序列比对问题[J]. 计算机工程与应用，2006，42(21)：38-42.

[11] Silva F J M, Pérez J M S, Pulido J A, et al. Parallel Niche Pareto AlineaGA—an evolutionary multiobjective approach on multiple

sequence alignment[J]. J Integr Bioinform, 2011, 8(3): 174.

[12] Blazewicz J, Frohmberg W, Kierzynka M, et al. G-MSA—A GPU-based, fast and accurate algorithm for multiple sequence alignment[J]. Journal of Parallel and Distributed Computing, 2013, 73(1): 32-41.

[13] 龚贺华. LSS-DCA：一个快速的分治多序列对齐算法[D]. 浙江大学，2003.

[14] 业宁，张倩倩，许翠云. 一种多序列比对分治算法DCA-ClustalW[J]. 计算机与数字工程，2010，38(11)：30-33.

下　篇

多序列比对参数篇

第 8 章　多序列比对的参数研究

8.1　基于 SP 目标函数的多序列比对参数研究

8.1.1　引言

多序列比对是生物信息学中最基本的工具，在序列分析、基因识别、蛋白质结构预测、生物进化树的构建等领域中有广泛应用，同时它也是一个 NP 完全问题，随着序列长度和条数的增多，时空复杂性急剧上升，如何设计一个具有高精度、高速度且低复杂度的多序列比对算法成为生物信息学中非常重要且具有挑战性的问题之一。Notredame 概括了构建多序列比对(MSA)算法的两大组成部分：①用来评估比对质量的目标函数；②用于识别所选目标函数最高分值(对应最优比对结果)的优化过程。

目标函数是用来考查多序列比对结果好坏的一种度量标准，所有的多序列比对方法都依赖于一个目标函数来说明比对结果的好坏，从而反映出此方法的精确度和有效性。当前有三种主流的目标函数：比对和函数(sum-of-pairs functions)、一致性函数(consensus functions)和树函数(tree functions)，其中使用最普遍的是比对和函数(简称为 SP 函数)。SP 函数需要设置两个重要的参数：替换矩阵(substitution matrix)和空位罚分(gap penalties，其中包括起始空位罚分和延续空位罚分)。Thompson 等根据序列的进化距离选用不同的替换矩阵，考虑亲水残基，提出特定残基位置的空位罚分。Gondro 认为空位罚分参数仍然凭经验给出，如何确定最佳的参数至今没有

理论框架。当前大多数文献的目标函数参数采用经验值。如果盲目选用一组目标函数参数，是否适合比对也未可知，直接用来比对也许会得到毫无意义的比对结果，造成无谓的浪费。本章基于 SP 目标函数，构建理论框架，从中推导出替换矩阵及空位罚分公式，根据待测序列的长度条数相似度等信息得到合适的替换矩阵与空位罚分，从而得到高质量的比对结果。

当前有很多学者根据多序列比对的原理开发了非常方便好用的开源在线比对工具，如 MAFFT、CLUSTALW、T-Coffee 等，应用这些比对工具能快速地得到较好的比对结果，成为当前多序列比对最常用的比对手段。但是这些结果对空位罚分与计分矩阵等参数的依赖性很强，不同参数下得到的结果很不一样，绝大多数用户在应用这些比对工具时使用单一的默认参数，这些默认参数虽然能得出较好的比对结果，但未必是最好的比对结果。另外，目前尚没有有效的方法直接确定最优参数值，故很难直接通过在线工具得到局部最优解。在各种常用的在线测试工具中，MAFFT 工具具有输入参数简单且比对结果较好等优点，本章以 MAFFT 作为基础实验工具以验证替换矩阵与空位罚分公式的正确性。

8.1.2　基本定义

1. 多序列比对问题及数学描述

一条长度为 k 的序列是 k 个字符组成的字符串，字符取自于字母表 {A, V, L, I, F, P, M, S, T, C, W, Y, N, Q, D, E, K, R, H, G}，分别代表蛋白质的二十个氨基酸残基类型。对于蛋白质序列，给定包含 N 个序列的序列集 $S=\{S_1,S_2,\cdots,S_N\}$，$N\geqslant 2$，$S_i=S_{i1}S_{i2}\cdots S_{il_i}(1\leqslant i\leqslant N)$，$S_{ij}\in\sum(1\leqslant j\leqslant l_i)$，$l_i$ 是第 i 条序列的长度，则一个序列比对可定义为一个矩阵 $\boldsymbol{A}=(a_{ij})$，其中 $1\leqslant i\leqslant N$，$1\leqslant j\leqslant l$，$\max(l_i)\leqslant l\leqslant\sum_{i=1}^{N}l_i$ 。

矩阵必须满足下列三个条件：

(1) $a_{ij} \in \sum \cup \{-\}$，其中“－”代表空位。
(2) 矩阵中的第 i 行去掉“－”后，即得到字符串 S_i。
(3) 矩阵中不包含字符全是空格的列。

2. 目标函数定义

sum-of-pair(SP)函数计算公式均可用 $score = \sum Residue - \sum penalty$ 表示。其中，score 定义为正数，分值越高，比对效果越好；$\sum Residue$ 是比对后的蛋白质序列中氨基酸残基的总分，定义为 $\sum Residue > 0$；$\sum penalty$ 是插入空格产生的总罚分，定义为 $\sum penalty > 0$。

氨基酸残基的总分公式为

$$\sum Residue = \sum_{h=1}^{L} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} Cost(S_i, S_j)$$

其中

$$Cost(S_i,S_j) = \begin{cases} S_{aa} = Score(a,a) & \text{如果两个残基相同(匹配)} \\ S_{ab} = Score(a,b) & \text{如果两个非空位残基不同(不匹配)} \end{cases}$$

目前常用的计分矩阵有两类：BLOSUM 矩阵和 PAM 矩阵。本章选用 BLOSUM 矩阵系列，如图 8.1 所示。

替换矩阵的 S_{aa} 互不相等，S_{ab} 也是互不相等的。一般来说，S_{aa} 的平均值 mean(S_{aa}) 和 S_{ab} 的最大值 max(S_{ab}) 可达到该数据的最基本要求。因此，为简化计算，以下统一规定 mean(S_{aa}) 为该矩阵匹配分值，max(S_{ab}) 为该矩阵非匹配分值。

其中，$\sum penalty$ 的计算根据插入空位的定义也分为两大类：线性罚分与仿射罚分。线性罚分是插入的每个空位罚同样的分数。仿射罚分根据生物定义将空位分为起始空位和延续空位，其产生的罚分分别简写为 GOP 和 GEP，$\sum penalty = N_{GOP} \cdot GOP + N_{GEP} \cdot GEP$，其中 N_{GOP} 是 GOP 的个数，N_{GEP} 是 GEP 的个数，且 $GOP>GEP$。具有生物意义的仿射罚分是当前最常应用的罚分方式。

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	**9**	-1	-1	-3	0	-3	-3	-3	-4	-3	-3	-3	-3	-1	-1	-1	-1	-2	-2	-2
S	-1	**4**	1	-1	1	0	1	0	0	0	-1	-1	0	-1	-2	-2	-2	-2	-2	-3
T	-1	1	**4**	1	-1	1	0	1	0	0	0	-1	0	-1	-2	-2	-2	-2	-2	-3
P	-3	-1	1	**7**	-1	-2	-1	-1	-1	-1	-2	-2	-1	-2	-3	-3	-2	-4	-3	-4
A	0	1	-1	-1	**4**	0	-1	-2	-1	-1	-2	-1	-1	-1	-1	-1	-2	-2	-2	-3
G	-3	0	1	-2	0	**6**	-2	-1	-2	-2	-2	-2	-2	-3	-4	-4	0	-3	-3	-2
N	-3	1	0	-2	-2	0	**6**	1	0	0	-1	0	0	-2	-3	-3	-3	-3	-2	-4
D	-3	0	1	-1	-2	-1	1	**6**	2	0	-1	-2	-1	-3	-3	-4	-3	-3	-3	-4
E	-4	0	0	-1	-1	-2	0	2	**5**	2	0	0	1	-2	-3	-3	-3	-3	-2	-3
Q	-3	0	0	-1	-1	-2	0	0	2	**5**	0	1	1	0	-3	-2	-2	-3	-1	-2
H	-3	-1	0	-2	-2	-2	1	1	0	0	**8**	0	-1	-2	-3	-3	-2	-1	2	-2
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	**5**	2	-1	-3	-2	-3	-3	-2	-3
K	-3	0	0	-1	-1	-2	0	-1	1	1	-1	2	**5**	-1	-3	-2	-3	-3	-2	-3
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	**5**	1	2	-2	0	-1	-1
I	-1	-2	-2	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	**4**	2	1	0	-1	-3
L	-1	-2	-2	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	**4**	3	0	-1	-2
V	-1	-2	-2	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	**4**	-1	-1	-3
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	**6**	3	1
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	**7**	2
W	-2	-3	-3	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	**11**

图 8.1　BLOSUM62 矩阵

8.1.3　公式推导

设待测序列共 m 条，最长序列长度为 $len_{\max}$，最短序列的长度为 $len_{\min}$，平均相似度为 $iden$，氨基酸残基匹配个数为 num_{match}，定义 $num_{match}=\dfrac{m(m-1)}{2}\cdot len_{\min}\cdot iden$。比对后每条序列插入空格的数目为 num_{gap}，在比对过程中不可能插入无限多的空位，通常规定空位数量不超过待测序列最长序列长度的 0.2 倍，即 $num_{gap}\leqslant \mathrm{INT}(0.2\cdot len_{\max})$，INT 是取整函数。例如，待测序列最大长度为 10，则每条序列最多插入 2 个空位。图 8.2 是比对前后的示意图。构建思路是取最优状态的待测序列(无空位)与其最差的比对结果(插入最多空位及最少匹配)进行比较，比对后的分数仍高于比对前的分数，此时得到的结果参数就是满足在比对过程所有情况下的参数。

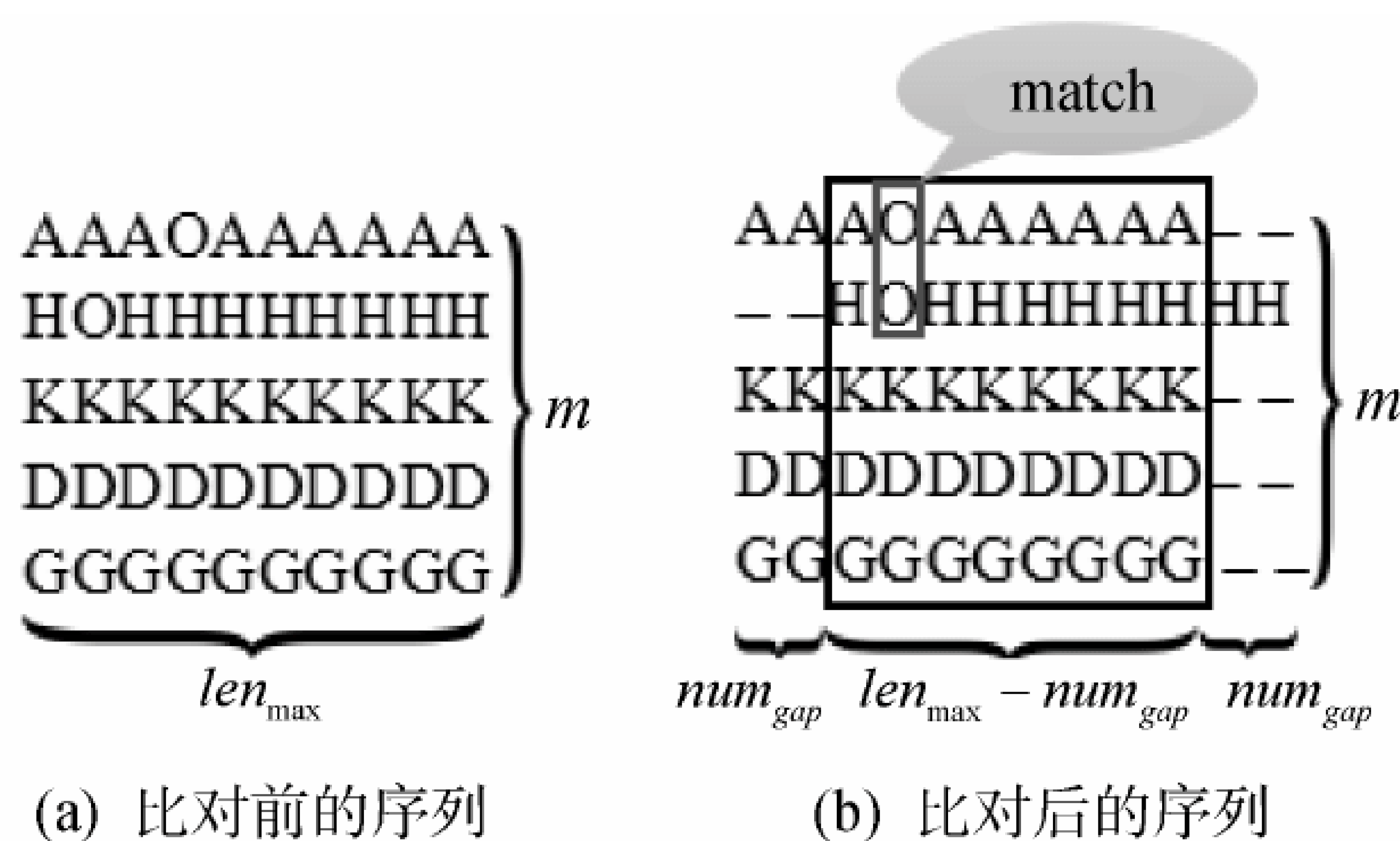


图 8.2　比对前后示意图

由 SP 公式计算得待测序列分数为

$$score_{\text{begin}} = \sum Residue - \sum penalty = \frac{m(m-1)}{2} \cdot len_{\max} \cdot S_{ab}$$

比对后的序列分数为

$$score_{\text{end}} = \left[\frac{m(m-1)}{2} \cdot (len_{\max} - num_{gap}) - num_{match} + \alpha \cdot \frac{(m-1)(m-2)}{2} \cdot num_{gap} \right] \cdot S_{ab} + \beta \cdot num_{match} \cdot S_{aa} - \sum penalty$$

从理论上，比对后的分数必须严格大于比对前的分数，即

$$score_{\text{begin}} < score_{\text{end}}$$

即

$$\frac{m(m-1)}{2} \cdot len_{\max} \cdot S_{ab} < \left[\frac{m(m-1)}{2} \cdot (len_{\max} - num_{gap}) - num_{match} + \alpha \cdot \frac{(m-1)(m-2)}{2} \cdot num_{gap} \right] \cdot S_{ab} + \beta \cdot num_{match} \cdot S_{aa} - \sum penalty \tag{8.1}$$

式中，α、β 是权重系数，其范围 $0 < \alpha < 1$，$0.8 < \beta < 1$。

1. 替换矩阵判断公式

式(8.1)简化为

$$\frac{(\alpha m-2\alpha-m)(m-1)}{2}\bullet num_{gap}\bullet S_{ab}+num_{match}\bullet(\beta S_{aa}-S_{ab}) \\ -\sum penalty>0 \qquad (8.2)$$

对式(8.2)再进行化简可得

$$S_{aa}>\left[\frac{(\alpha m-2\alpha-m)(1-m)}{2\beta}\bullet\frac{num_{gap}}{num_{match}}+\frac{1}{\beta}\right]\bullet S_{ab}$$

可得替换矩阵判断公式

$$mean(S_{aa})>\left[\frac{(\alpha m-2\alpha-m)(1-m)}{2\beta}\bullet\frac{num_{gap}}{num_{match}}+\frac{1}{\beta}\right]\bullet\max(S_{ab}) \qquad (8.3)$$

假设 $mean(S_{aa})$ 为参考值 $reference$，$\left[\frac{(\alpha m-2\alpha-m)(1-m)}{2\beta}\bullet\frac{num_{gap}}{num_{match}}+\frac{1}{\beta}\right]\bullet\max(S_{ab})$ 为计算值 $calc$，则式(8.3)可简化为

$$reference>calc \qquad (8.4)$$

根据式(8.4)可判断所选取的计分矩阵是否合理。

例如 451c_ref1 序列，共 5 条，最大长度是 87，最小长度是 80，相似度是 23%，欲选取 BLOSUM45 矩阵，其 $\max(S_{ab})=3$，$\alpha=0.5$，$\beta=0.95$，则其计算值为

$$calculated\quad value=\left[\frac{(\alpha m-2\alpha-m)(1-m)}{2\beta}\bullet\frac{num_{gap}}{num_{match}}+\frac{1}{\beta}\right]\bullet\max(S_{ab}) \\ =5.492$$

其参考值 $reference=mean(S_{aa})=7.05$，符合式(8.4)要求，即可以选择 BLOSUM45 矩阵作为 451c_ref1 的替换矩阵。

2. 空位罚分公式

设待测序列有 m 条序列，比对后每条序列插入空格的数目为 num_{gap}，在仿射罚分前提下，假设每条序列中空位数是起始空位的 λ

倍，则 $N_{GOP}=m\bullet\frac{1}{\lambda}\bullet num_{gap}$，$N_{GEP}=m\bullet\left(1-\frac{1}{\lambda}\right)\bullet num_{gap}$，因为 $GOP>GEP$，设 $GOP=n\bullet GEP$，其中 λ、n 是正整数，则

$$\sum penalty=N_{GOP}\bullet GOP+N_{GEP}\bullet GEP=\frac{n+\lambda-1}{n\lambda}\bullet m\bullet num_{gap}\bullet GOP$$

由式(8.2)的简化有

$$\frac{(\alpha m-2\alpha-m)(m-1)}{2}\bullet num_{gap}\bullet S_{ab}+num_{match}\bullet(\beta S_{aa}-S_{ab})>\sum penalty\Rightarrow$$

$$GOP<\left[\frac{(\alpha m-2\alpha-m)(m-1)}{2}\bullet num_{gap}\bullet S_{ab}+num_{match}(\beta S_{aa}-S_{ab})\right]\bullet\frac{n\lambda}{m(n+\lambda-1)\bullet num_{gap}} \tag{8.5}$$

式(8.5)为起始空位罚分 GOP 的上限计算公式，且其下限 $GOP>0$。

如果在 GOP 上限乘以权重系数 $0<\omega<1$，可得最佳 GOP 估计公式：

$$GOP=\omega\bullet\left[\frac{(\alpha m-2\alpha-m)(m-1)}{2}\bullet num_{gap}\bullet S_{ab}+num_{match}(\beta S_{aa}-S_{ab})\right]\bullet\frac{n\lambda}{m(n+\lambda-1)\bullet num_{gap}} \tag{8.6}$$

式中，

$$num_{match}=\frac{m(m-1)}{2}\bullet len_{\min}\bullet iden\text{，}\quad num_{gap}=\mathrm{INT}(0.2\times len_{\max})$$

INT 是取整函数，$len_{\min}$ 是待测序列中最短序列的长度，$iden$ 是待测序列的平均相似度。

最佳 GEP 估计公式：

$$GEP=GOP/n \tag{8.7}$$

式(8.6)与式(8.7)中的各权重系数 α、β、λ、n、ω 将通过实验数据验证给出最佳数值。

8.1.4　实验结果与分析

1. 实验设计

为了对数据进行统一且精确的比较，本节以 BAliBASE2.0 数据库 ref1～ref3 共计 113 组序列簇作为测试对象，且应用该数据库的 SPS(sum of pair score)作为统一的比对评价标准，*SPS* 分值表示残基对准确对齐的比率，*SPS* 值越高，说明比对的结果越接近于参考序列，比对效果越好。

当前有很多热门的在线比对工具，如 MAFFT、ClustalW、T-Coffee、MUSCLE 等，它们的比对速度都很快，比对结果也相差无几，之所以选择 MAFFT 作为比对工具有以下几点原因：①需要输入的参数极简，只需输入替换矩阵、起始空位罚分 *GOP* 和延续空位罚分 *GEP* 三个参数即可，其中替换矩阵只有 BLOSUM 系列的三个矩阵；②可以批量比对；③比对结果相对优于其他在线工具。实验数据设置空位罚分范围分别为$1 \leqslant GOP \leqslant 20$，$0 \leqslant GEP \leqslant \dfrac{GOP}{2}$，且 *GOP* 步长为 1，*GEP* 步长为 0.2，替换矩阵分别为 BLOSUM30/BLOSUM45/BLOSUM62，每一组序列有 1590 个互不相同的参数组合，通过批量处理，得到全面的比对结果作为参考数据库。图 8.3 将比对结果以三维图呈现出来，可以看出 *SPS* 值与 *GOP*/*GEP*/*MATRIX* 相关，当 GOP/GEP/MATRIX 取特定值时，*SPS* 达到最大值，即该序列比对效果最好。

2. 实验验证

1) 实验 1：矩阵公式合理性的验证

根据式(8.3)判断替换矩阵是否适合待测序列，图 8.4 所示是 ref1～ref3 的三个矩阵判断图。其中，直线表示该矩阵的参考值，点连线表示该矩阵的计算值，由式(8.4)知，当点连线位于直线下方时，该矩阵符合序列比对要求，若点连线在直线上方时，该矩阵不

能用于序列比对。BLOSUM45 和 BLOSUM62 的 $\max(S_{ab})$ 相等，所以它们的计算值是重合的。通过图 8.4 可以看出 BLOSUM30 不能满足大多数序列的比对要求，BLOSUM45 和 BLOSUM62 基本满足比对要求。

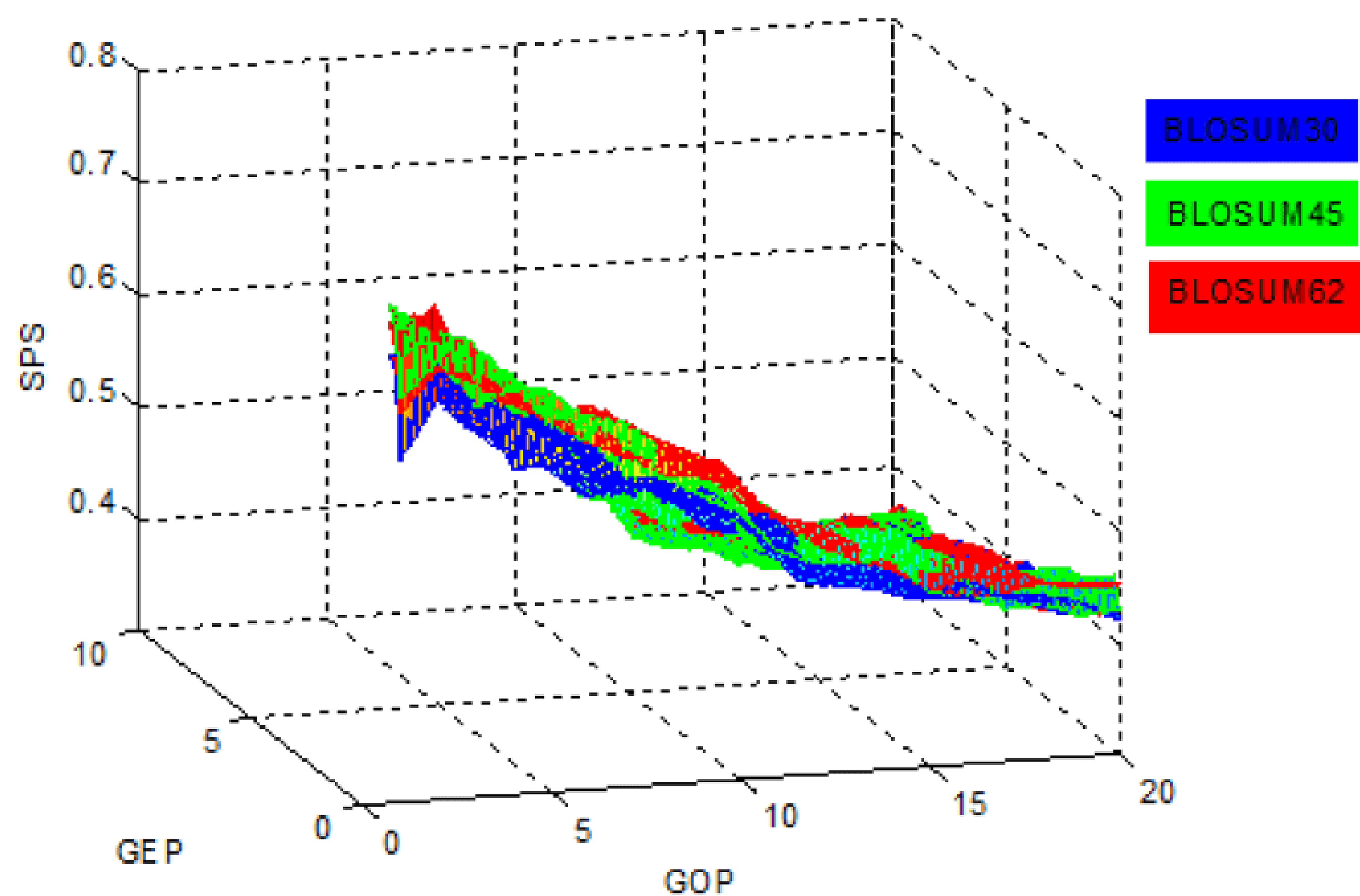


图 8.3　序列比对 SPS/GOP/GEP/MATRIX 三维图

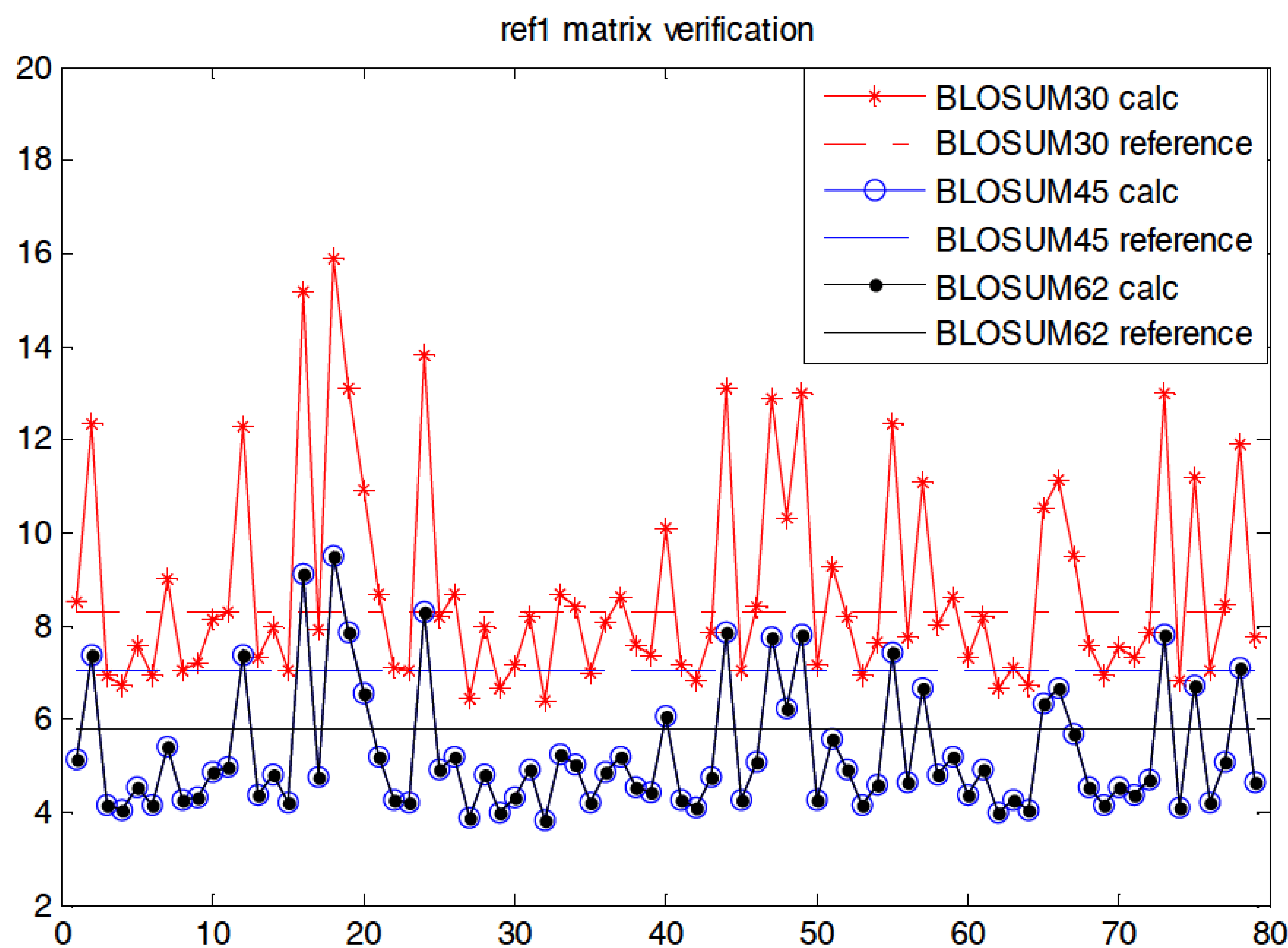


图 8.4　替换矩阵合理性的判断图

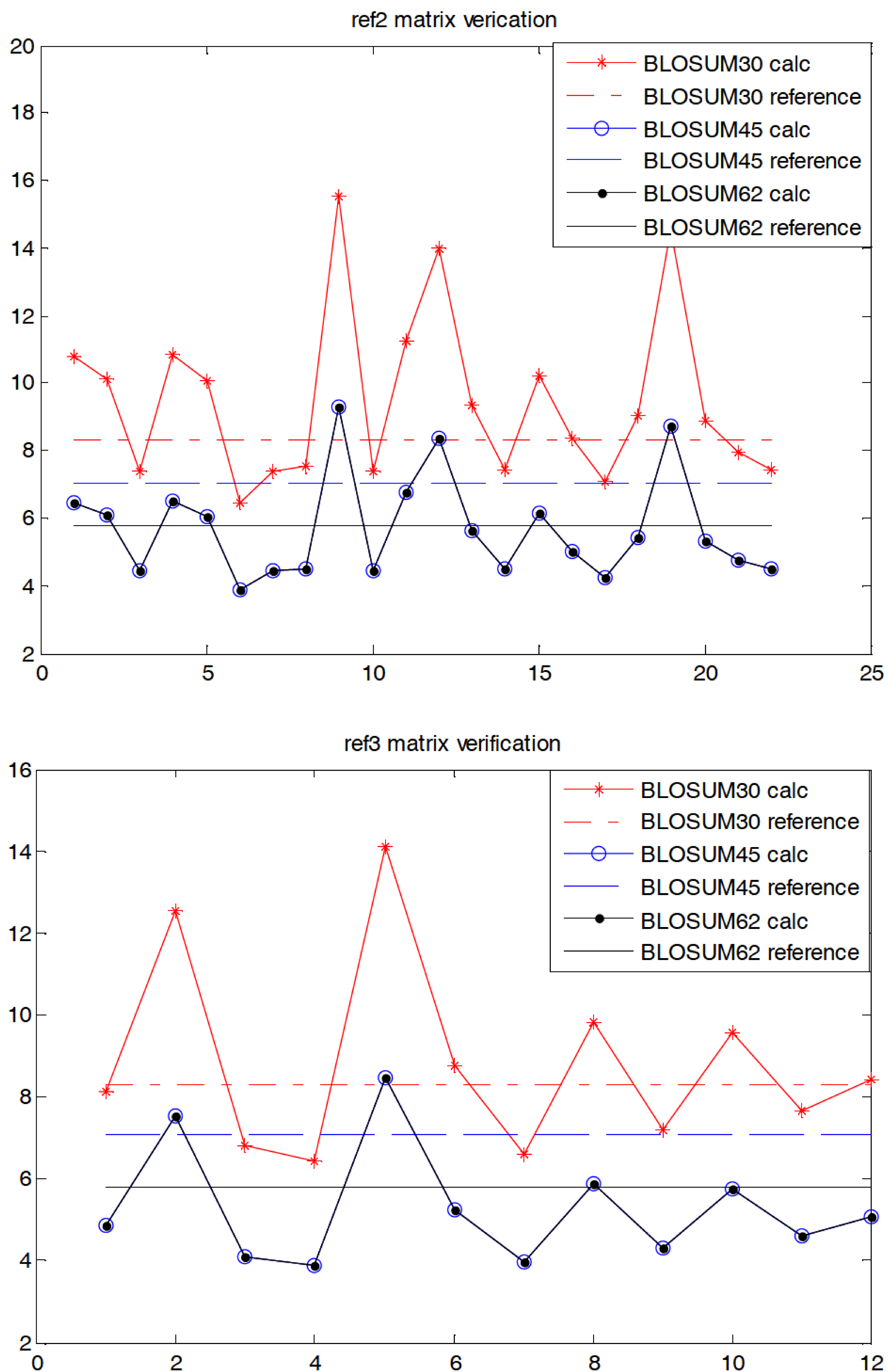


图 8.4　替换矩阵合理性的判断图(续)

在表 8.1 中统计了图 8.4 中三个替换矩阵符合比对要求的次数，可以看出 BLOSUM45 能满足绝大多数序列的比对要求，因此本书以 BLOSUM45 作为最佳替换矩阵，以下所有实验数据均以 BLOSUM45 为准。

表 8.1　替换矩阵次数统计表

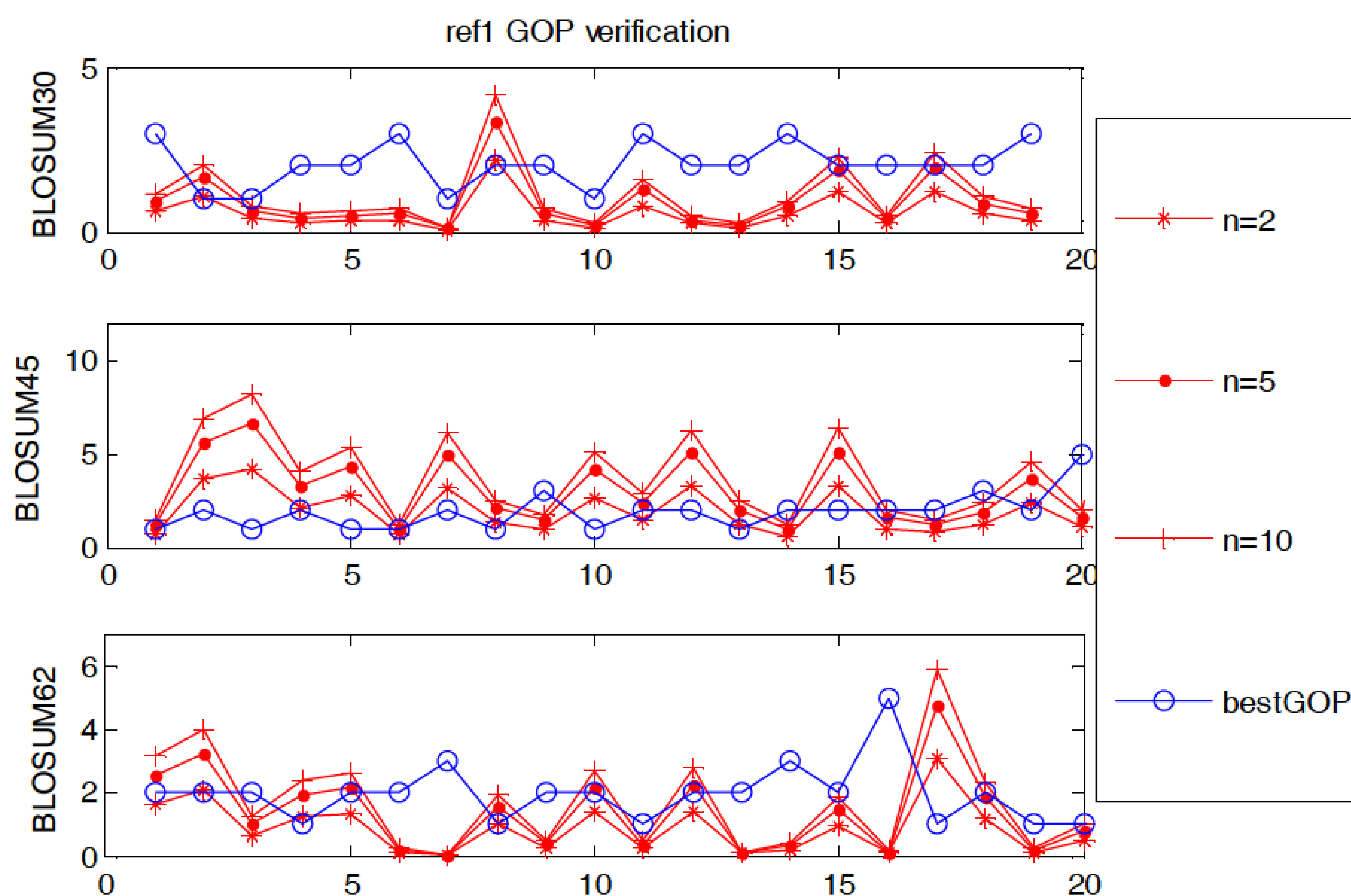
Sequence set	ref1-Test1	ref1-Test2	ref1-Test3	ref2	ref3
Sequence Length(bp)	<100	100～300	>300	50～600	60～600
Sequence number	27	24	28	22	12
BLOSUM30 Qualified number	10	8	7	2	4
BLOSUM45 Qualified number	**21**	**19**	**21**	**13**	**10**
BLOSUM62 Qualified number	20	14	18	9	6
Correct rate	78%	79%	75%	60%	83%
最佳替换矩阵	BLOSUM45				

2) 实验 2：GOP 公式的验证

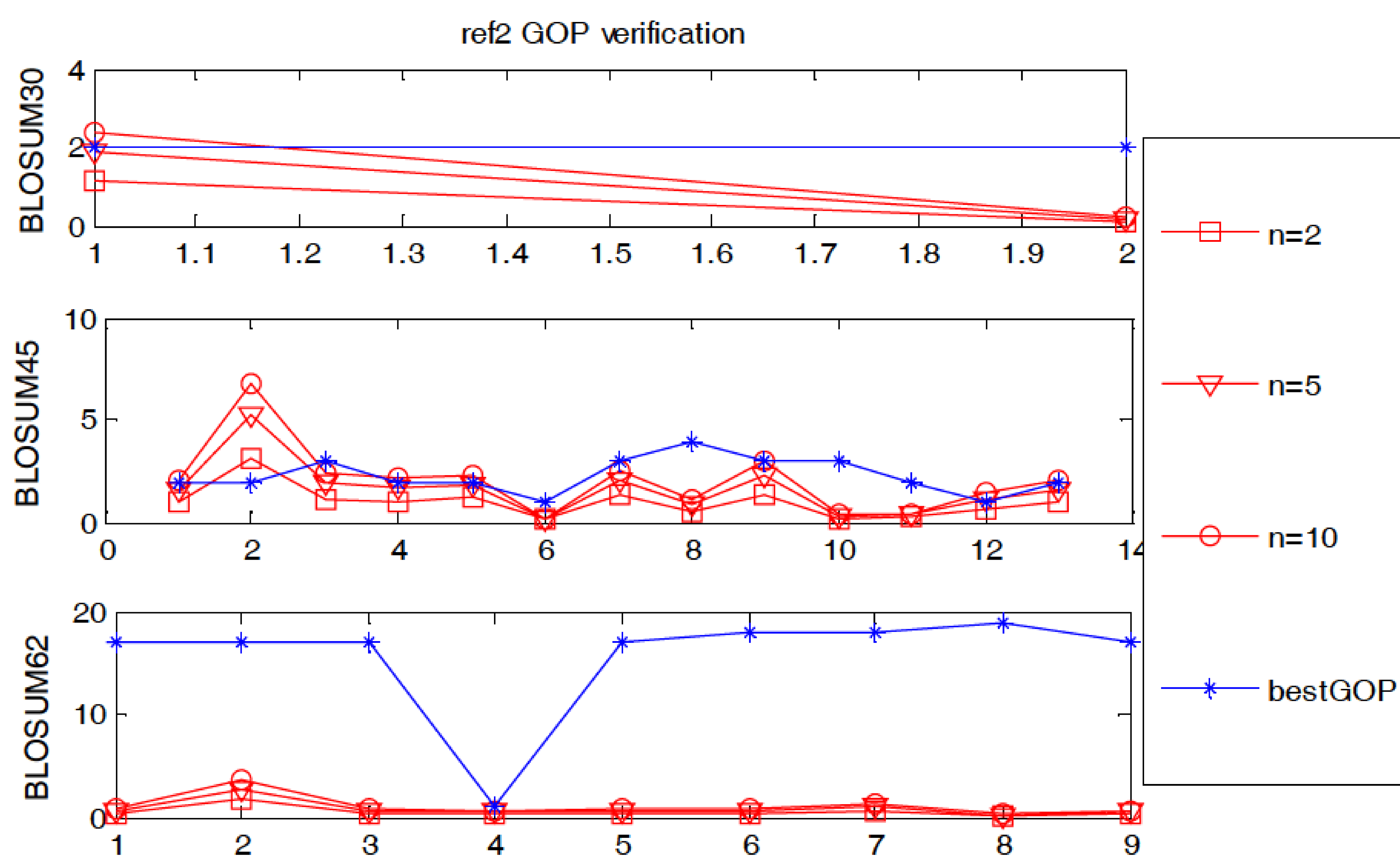
对于空位罚分的取值公式(8.6)和(8.7)进行验证，如图 8.5 所示，其中最佳 GOP 是图 8.3 中最大 SPS 对应的 GOP 值。从图 8.5 可以看出：

(1) 当替换矩阵为 BLOSUM45 矩阵时，计算的 GOP 曲线与最佳 GOP 曲线最为贴近，即 BLOSUM45 计算效果最好。

(2) 当 $GEP = GOP/5$ 即 n=5 时，GOP 计算值与最佳 GOP 值最为贴近。

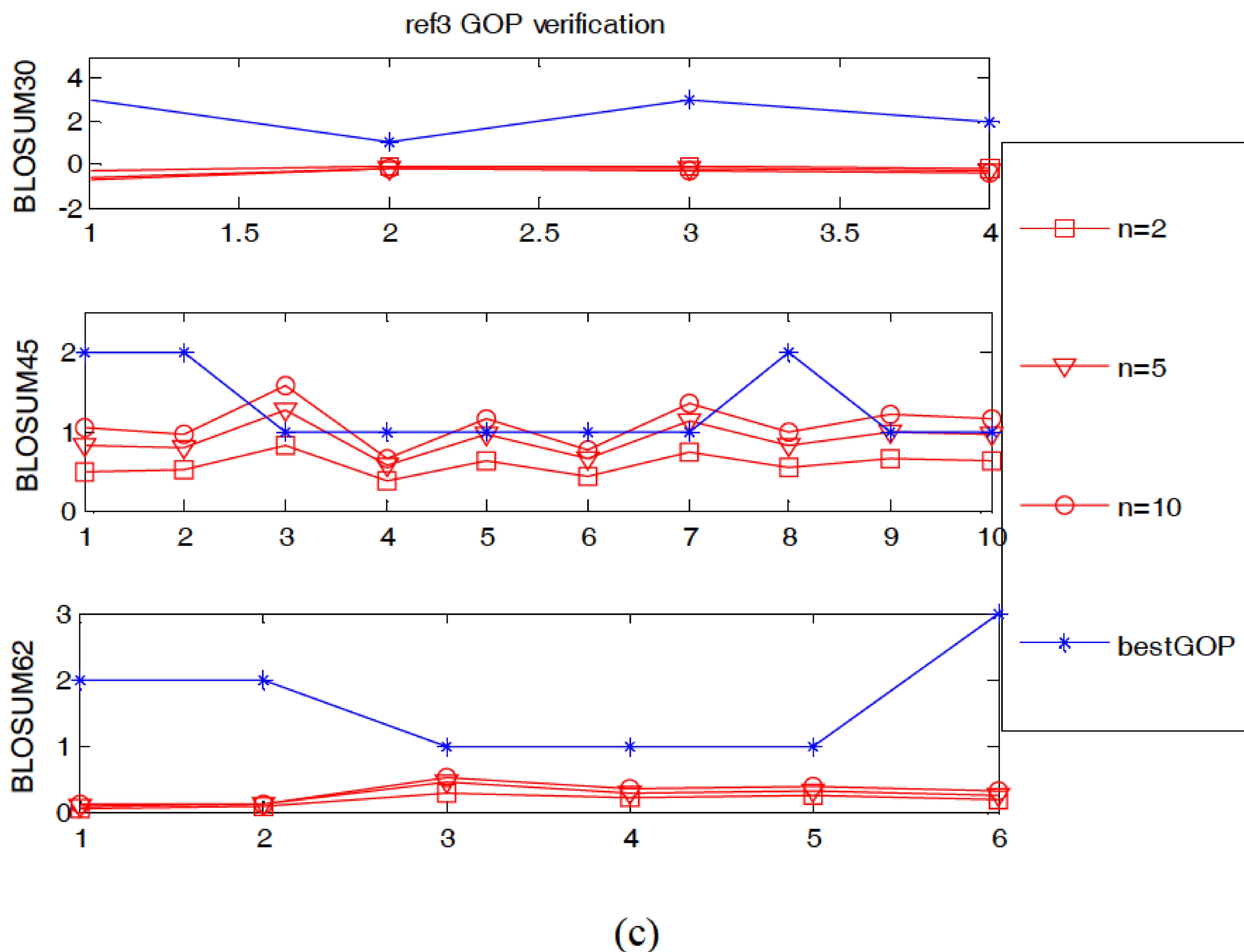


(a)



(b)

图 8.5 GOP/GEP 公式验证图



(c)

图 8.5　GOP/GEP 公式验证图(续)

3) 实验 3：最优参数合理性的验证

根据上面两个实验可以得到一组固定参数：$\alpha=0.5, \beta=0.9, n=5$，替换矩阵是 BLOSUM45。其他的参数与序列相关，其中，λ 是每条序列中起始空位个数与总空位数 num_{gap} 的比例，空位个数 $num_{gap}=\mathrm{INT}(0.2 \cdot len_{\max})$，所以 λ 与该序列的长度有关。通过数据分析，发现当序列长度<100bp 时(定义为短序列)，$\lambda=3$；当 100bp<序列长度<300bp 时(定义为中序列)，$\lambda=4$；当序列长度>300bp 时(定义为长序列)，$\lambda=5$，即序列长度越长，λ 越大，这样的设定可以取得最好的效果。这个规律表示当序列长度增长时，起始空位的个数增长受到一定制约，不会增加得太多，连续空位分布较为集中，更符合多序列比对的生物特性。其他的参数也是根据能达到最大 SPS 值的要求设定的。在表 8.2 中整理了 ref1～ref3 的最优权重系数，并附上序列信息。根据表 8.2 中的最优权重系数计算出最优参数 GOP/GEP/MATRIX，从图 8.3 的参考数据库中找出对应的 SPS 值，

并与 MAFFT 和 CLUSTALW 默认参数值算出的 SPS 值进行对比，曲线对比效果见图 8.6。

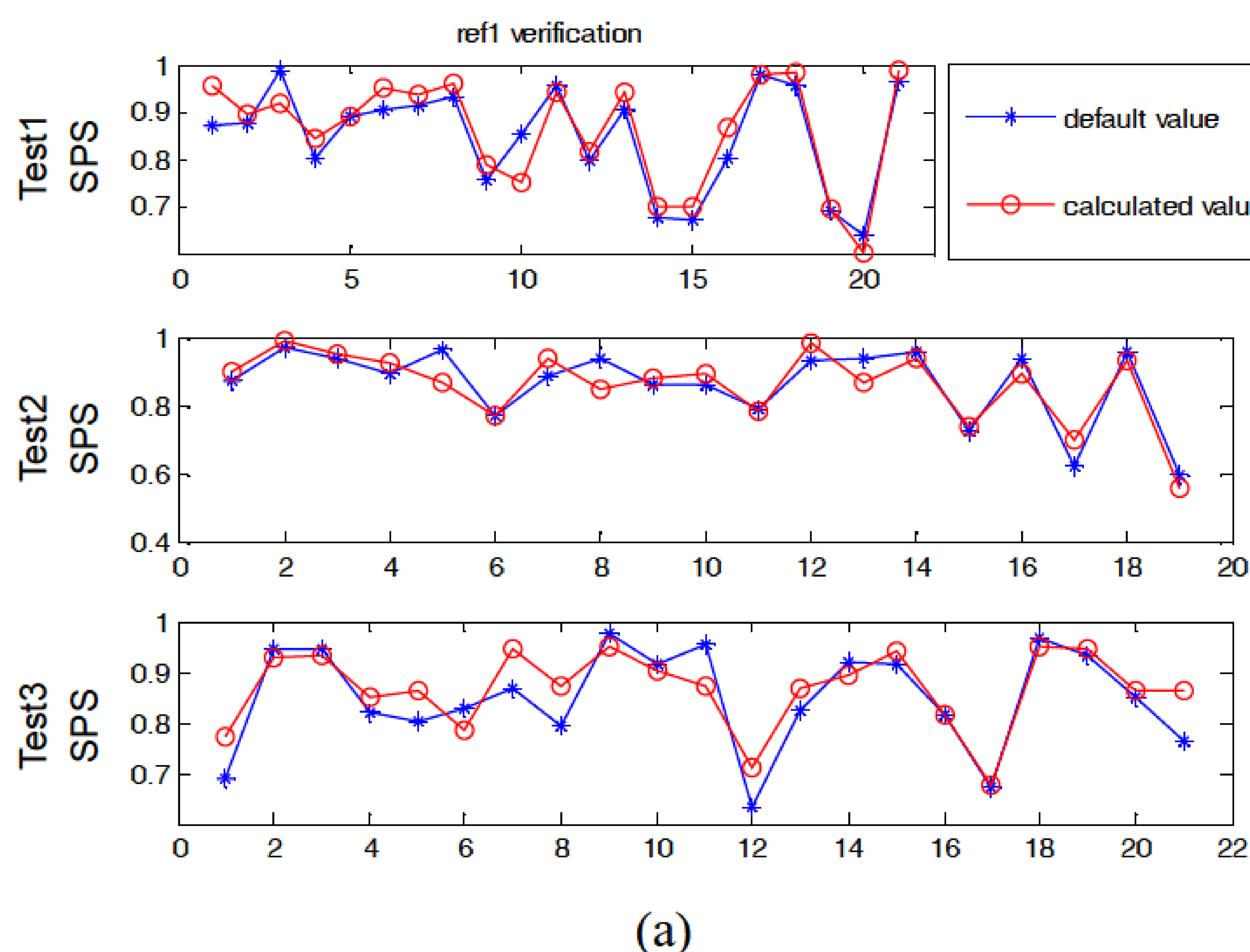
通过表 8.2 和图 8.6 可以得到以下结论：

(1) 根据序列长度，ref1 分为短中长三个 test，λ 的取值与序列长度密切相关，ref2 和 ref3 的序列长度没有明显分类，有短序列，也有中长序列，故在计算 ref2 和 ref3 时根据其具体的序列长度设定 λ。

(2) 将最优参数对应的 SPS 与 MAFFT 默认参数的 SPS 值进行横向比较，最优参数所得的 SPS 值绝大多数超过默认值，说明最优参数对多序列比对有优化作用。

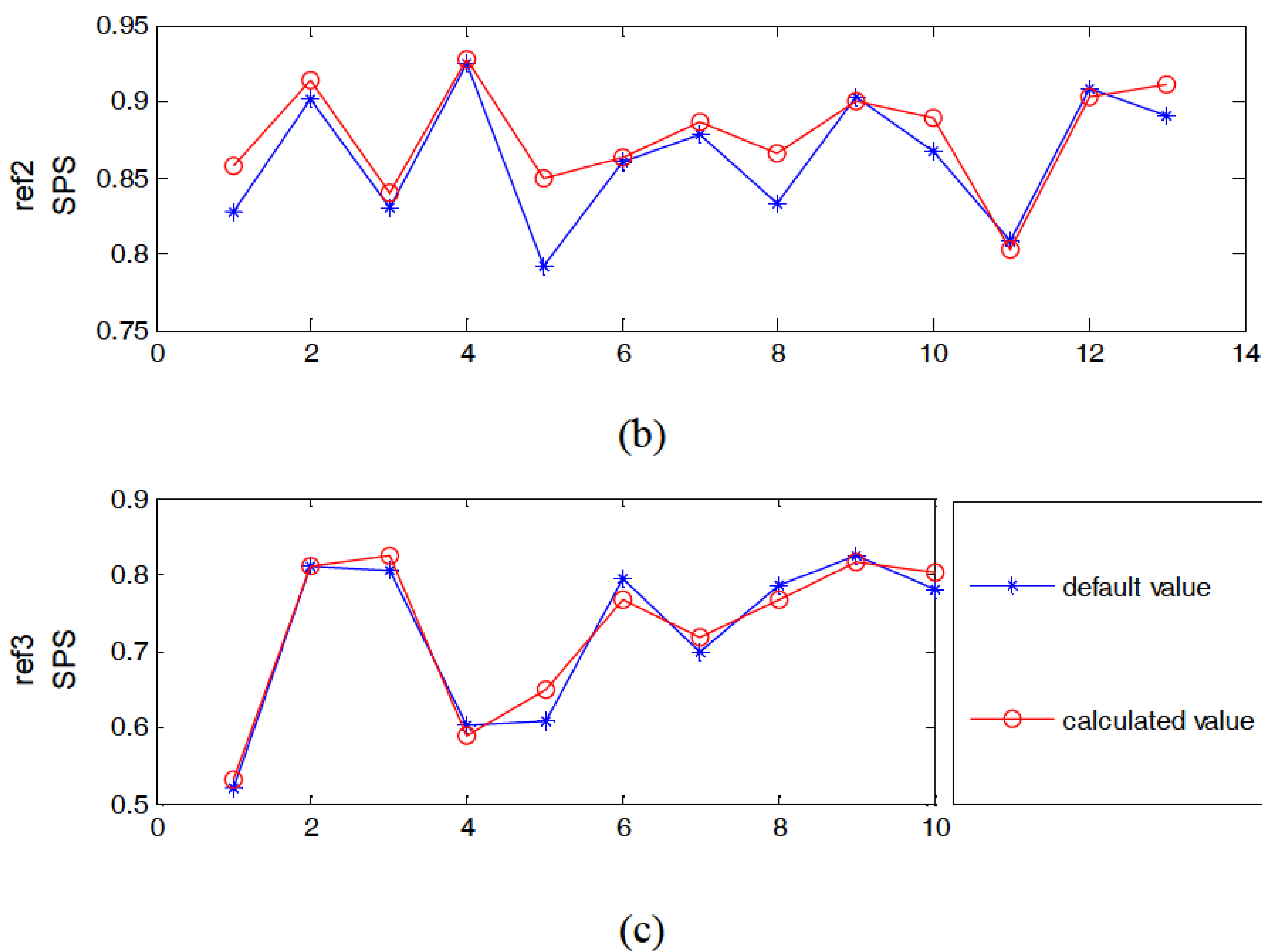
表 8.2　最优权重系数表

Sequence set	Sequence row	Sequence Length(bp)	Sequence number	Qualified number	Correct number	Correct rate	λ	ω
ref1 test1	4～5	<100	27	21	17	81%	3	0.1
ref1 test2	4～5	100～300	24	19	11	58%	4	0.3
ref1 test3	4～5	>300	28	21	13	62%	5	0.2
ref2	14～19	50～600	22	13	10	77%	与长度相关	0.05
ref3	>20	60～600	12	10	6	60%	与长度相关	0.02



(a)

图 8.6　最优参数计算值与 MAFFT 默认值的比较图



(b)

(c)

图 8.6 最优参数计算值与 MAFFT 默认值的比较图(续)

8.1.5 结论

本节应用 MAFFT 工具解决多序列比对问题，为了得到更好的比对结果，在比对过程中摒弃常用的默认参数，致力于寻找一组最优参数。基于 SP 目标函数，提出了明确的目标函数参数理论依据，给出替换矩阵的判断公式和空位罚分最佳取值公式，并应用 BAliBASE2.0 数据库实例从替换矩阵、空位罚分和与默认值横向比较这三个角度验证公式的合理性。实例证明最终的比对结果强烈依赖于 SP 目标函数的参数包括替换矩阵和空位罚分，一组好的参数会得到一个好的比对结果。根据待测序列的长度条数相似度等信息代入本节公式可以得到最合适的起始空位罚分、延伸空位罚分和替换矩阵。实验结果表明，通过本节的公式，应用 MAFFT 工具，可以获得更高的精度校准，并获得更优质的比对结果。这项研究将优化多序列比对算法，并提供多序列比对的新思路。

在今后的工作中，还有一些需要完善的地方：

(1) 可以考虑应用其他数据库的数据来测试本节公式的合理性与通用性。

(2) 可以应用本节思路寻找其他比对工具的最佳参数。注意，本节所得的权重参数都是基于 MAFFT 下产生的数据，因此该组参数仅适用于 MAFFT 比对工具，对于其他比对工具不一定通用。

(3) 可以考虑其他的目标函数和其他的替换矩阵，如 PAM 矩阵或 GONNET 矩阵。

(4) 可以推导 *GOP* 和 *GEP* 更合适的关系式，本节中关于 *GOP* 和 *GEP* 的关系定位较为粗糙，仅根据 *GOP*>*GEP* 定位为倍数关系，事实上，应该会有别的更恰当的关系式有待推导。

希望通过以上的思路提高公式的通用性，更进一步提高多序列比对的质量。

8.2　在线工具 MAFFT 参数研究

8.2.1　引言

多序列比对是生物信息学中最基本的应用工具，其在蛋白质结构预测分析、基因识别、构建生物进化树等领域中都有广泛的应用。它是一个 NP 完全问题，随着序列长度和条数的增多，时空复杂性急剧上升，如何设计一个具有高精度、高速度且低复杂度的多序列比对算法成为生物信息学中非常具有挑战性的一个重要课题。

多序列比对有两个重要的参数：替换矩阵和空位罚分(包括起始空位和延续空位)。有很多学者相继讨论过这些参数，如 Thompson 等根据序列的进化距离选用不同的替换矩阵，考虑亲水残基，提出特定残基位置的空位罚分(1994)；Reese, J. T 和 Pearson, W. R (2002) 讨论了多序列比对中 PAM 矩阵的 PAM 距离与空位罚分的关系式；

Madhusudhan, M. S 等(2006)应用动态规划算法根据序列结构提出可变罚分公式 VGP。但是这些公式并没有被广泛应用，说明它们不具有通用性，Gondro, C 和 Kinghorn, B. P(2007)认为空位罚分参数仍然凭经验给出，如何确定最佳的参数至今没有理论框架，因此当前大多数文献的目标函数参数仍然采用经验值(Dan, D. B 和 Kececioglu, J. 2015)。

当前有很多学者根据多序列比对的原理开发了非常方便好用的开源在线比对工具，如 CLUSTALW、T-Coffee、MAFFT 等(Katoh, K 和 Toh, H, 2008)，应用这些比对工具能快速得到较好的比对结果，成为当前多序列比对最常用的比对方式。但是，这些结果对空位罚分与计分矩阵等参数的依赖性很强，不同参数下得到的结果不一样，绝大多数用户在应用这些比对工具时使用单一的默认参数，这些默认参数虽然能得出较好的比对结果，但未必是最好的比对结果。另外，目前尚没有有效的方法直接确定最优参数值，故很难直接通过在线工具得到局部最优解。Pais, S. M 等(2014)总结了各种常用的多序列比对方法和工具的比对效率，如 CLUSTALW、CLUSTAL OMEGA、DIALIGN-TX、MAFFT、MUSCLE、POA、Probalign、ProbCons 和 T-Coffee，认为 T-Coffee 和 MAFFT 可以更快速高效地比对序列。Nuin, P. A、Wang, Z 和 Tillier, E. R (2006)比较这 9 种常用比对工具：CLUSTALW、Dialign2.2、T-Coffee、POA、MUSCLE、MAFFT、ProbCons、Dialign-T 和 Kalign，并得出下面结论：9 种比对软件中，MAFFT 的迭代方法(L-INS-i)和 ProbCons 工具始终是最准确的，并且 MAFFT 是两者中较快的比对工具。Ahola, V 等(2006)提出了基于一种统计分数来评估多序列比对的结果，以 BAliBASE 作为标准数据库，比较了 7 种比对方法的 AQ 得分，结果表明，MAFFT 的 L-INS-i 方法优于其他方法。他们的权威结论都被归纳在 MAFFT 官方网页中。多序列比对的计算效率需要综合考虑速度和精度，MAFFT 设定 FFT-NS-2 为默认迭代算法，然而，随着计算机科学的高速发展，多序列比对的重心也从追求高速度转为追求高精度，因

此本研究以 MAFFT 作为比对工具，试图寻找替换矩阵、空位罚分及迭代算法的最优参数组合，以得到高精度的比对结果。

通常使用手工或半自动的序列数据库评估比对工具的精确度，如 BAliBASE(Thompson, J. D 等，2005)、PREFAB 和 SABmark (Walle, I. V 等，2004)。目前为止，BAliBASE 是最普遍使用的数据库，它是基于已知的三维结构的蛋白质序列和模型来构造的序列数据库。本研究选用最新版本的 BAliBASE3.0 数据库作为实验测试对象和评价标准。

8.2.2　基本定义

1. 多序列比对问题及数学描述

二条长度为 k 的序列是 k 个字符组成的字符串，字符取自于字母表{A, V, L, I, F, P, M, S, T, C, W, Y, N, Q, D, E, K, R, H, G}，分别代表蛋白质的二十个氨基酸残基类型。对于蛋白质序列，给定包含 N 个序列的序列集 $S=\{S_1,S_2,\cdots,S_N\}$, N=2，$S_i=S_{i1}S_{i2}\cdots S_{il_i}(1\leqslant i\leqslant N)$，$S_{ij}\in\sum(1\leqslant j\leqslant l_i)$，$l_i$ 是第 i 条序列的长度，则一个序列比对可定义为一个矩阵 $\boldsymbol{A}=(a_{ij})$, 其中 $1\leqslant i\leqslant N, 1\leqslant j\leqslant l$，$\max(l_i)\leqslant l\leqslant\sum_{i=1}^{N}l_i$ 。

且矩阵必须满足下列三个条件：

(1) $a_{ij}\in\sum\cup\{-\}$，其中“－”代表空位。

(2) 矩阵中的第 i 行去掉“－”后，即得到字符串 S_i。

(3) 矩阵中不包含字符全是空格的列。

2. MAFFT 比对工具

MAFFT 比对工具起初是为了执行大规模序列比对而发展起来的，它是一种基于快速傅里叶变换(FFT)的组对组的比对算法，并且它使用一种近似距离计算方法(6mer 方法)便于进行快速计算。为了提高多

序列比对结果的精度，以及随着序列距离相似度的增加，2005 年，发布了 MAFFT 版本 5，2008 年和 2013 年相继发布了 MAFFT 版本 6 和 MAFFT 版本 7。

MAFFT 提供了多种多序列比对策略，这些比对策略可以划分成三大类：渐进方法(the progressive method)、基于 WSP 分数的迭代细化方法(the iterative refinement method with the WSP score)和基于 WSP 和一致性分数的迭代细化方法(the iterative refinment method using both the WSP and consistency scores)，如表 8.3 所示。通常，MAFFT 策略需要在速度和精度之间折中考虑，上述三类方法的速度依次递减，然而精度却依次递增。

MAFFT 的默认参数如下：使用的算法是 FFT-NS-2，GOP 是 1.53，GEP 是 0.123，矩阵是 BLOSUM62。

表 8.3　MAFFT 算法分类及说明

类别	算法名称	说明
渐进方法	FFT-NS-1	速度比默认参数快 2 倍
	FFT-NS-2	默认参数
基于 WSP 分数的迭代细化方法	FFT-NS-I	在渐进方法和基于 WSP 分数的迭代细化方法类别中速度最快的算法，使用 WSP 打分方法
	NW-NS-I	使用 6mer 方法计算序列之间的距离
基于 WSP 和一致性分数的迭代细化方法	L-INS-I	使用 WSP 和一致性打分方法进行局部比对
	E-INS-I	基于广义仿射空位罚分使用 WSP 和一致性打分方法进行局部比对基于广义仿射空位罚分
	G-INS-I	使用 WSP 和一致性打分方法进行全局比对

8.2.3　实验结果与分析

1. 实验设计

为了对数据进行统一且精确的比较，本节以 BAliBASE3.0 数据库 ref1～ref3 共计 113 组序列簇作为测试对象，因为该数据库中的参考序列均是手工比对，结果更具有生物特性，成为测试算法的常用数据库之一，见表 8.4。应用该数据库的 SPS 打分函数作为统一的比对评价标准，SPS 分值表示残基对准确对齐的比率，SPS 值越高，说明比对的结果越接近于参考序列，比对效果越好。

表 8.4　BAliBASE3.0 中的数据集

数据集	RV11	RV12	RV20	RV30	RV40	RV50	合计
序列的个数	38	45	41	30	48	16	218

实验数据设置：MAFFT 在线工具的 GOP 范围为 0～3，GEP 可以自由输入。本节为提高计算效率，设置空位罚分范围分别为 $0.5 \leqslant GOP \leqslant 3$，$0.03 \leqslant GEP \leqslant \frac{GOP}{2}$，且 GOP 步长为 0.1，GEP 步长为 0.03，在 BLOSUM30/BLOSUM45/BLOSUM62/BLOSUM80/PAM100/PAM200 中选择替换矩阵，每一组序列有 692 个互不相同的参数组合，通过批量处理，得到全面比对结果的 SPS 值作为参考数据库。图 8.7 将所有的比对结果以三维图呈现出来。可以看出，SPS 值与 GOP/GEP/MATRIX 相关，当 GOP/GEP/MATRIX 取特定值时，SPS 达到最大值，即该序列比对效果最好。

2. 实验验证

1) 实验 1：MAFFT 迭代算法及替换矩阵的筛选

根据表 8.3 关于 MAFFT 算法的介绍，这里选择精度最高的三种迭代算法 L-NS-I/E-NS-I/G-NS-I。

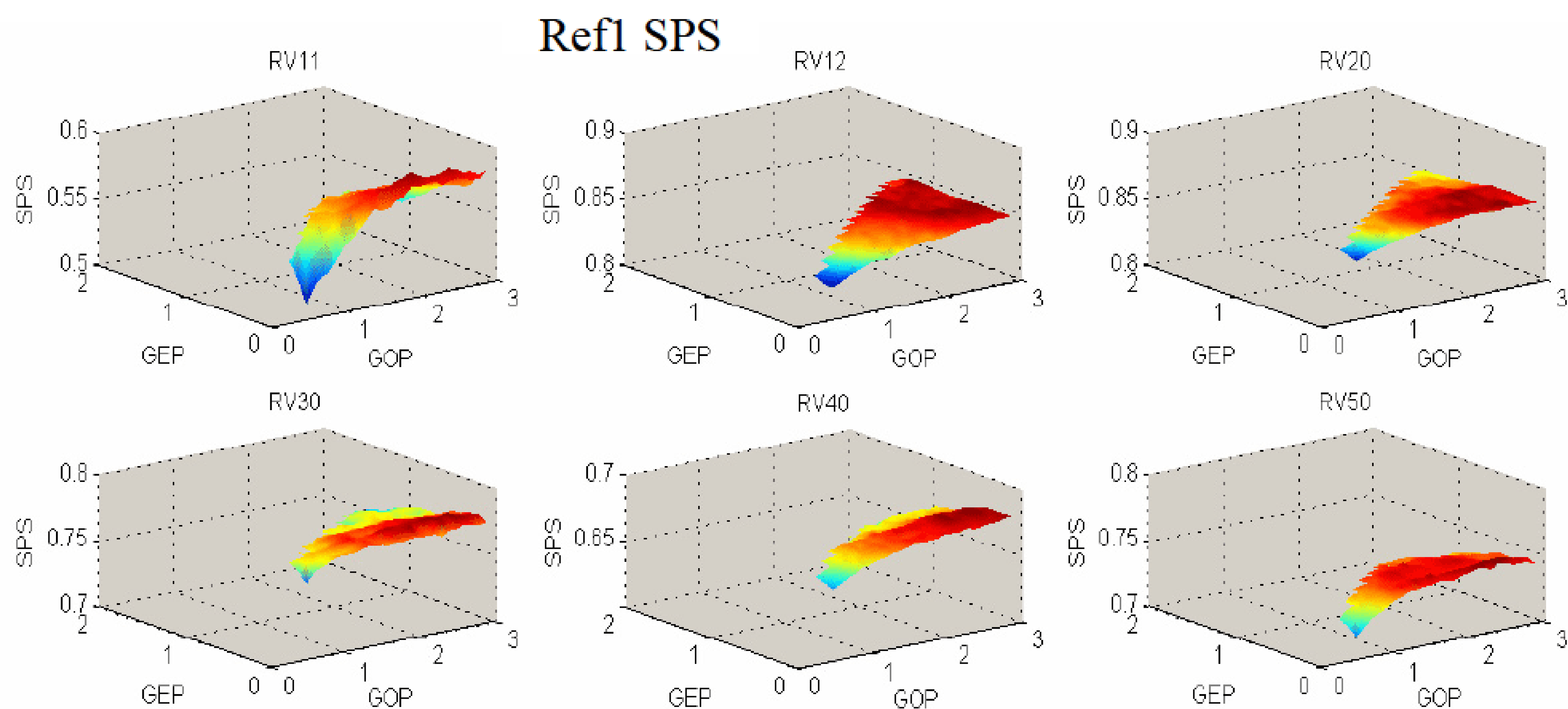


图 8.7　SPS 函数值(基于 GOP/GEP/MATRIX)

筛选思路如下：

(1) 对于 BAliBASE3.0 的每一组序列，分别计算三种迭代算法六种替换矩阵在默认值 *GOP*=1.53，*GEP*=0.123 的 SPS。

(2) 每一种组合下的 SPS 平均值命名为 meanSPS，例如 RV11 共有 67 组序列簇，当比对参数为：L-NS-I/BLOSUM30/GOP=1.53/GEP=0.123 时，可以得到 67 个比对结果，即有 67 个 SPS 值，计算 SPS 的平均值为 meanSPS=0.5132。其他数值以此类推。

(3) 统计每一种迭代算法下 meanSPS 的总和，则每一组序列可以得到三个值，这三个值的最大值对应的迭代算法认为是最优算法。

(4) 在每一种迭代算法下六个矩阵分别比对，因此得到六组 SPS 值，定义达到最大 SPS 值的个数为 maxSPSnum。例如 RV11 共有 67 组序列簇，在 L-NS-I 迭代算法时，每一组序列都有 6 个 SPS 值，分别对应于 6 个替换矩阵，统计取得 SPS 最大值所对应的矩阵个数，67 组序列中有 9 组序列以 BLOSUM30 为替换矩阵的 SPS 比其他矩阵的 SPS 值高，则 maxSPSnum=9。

(5) 从最优的迭代算法中选择前两名 maxSPSnum 对应的矩阵作为最优矩阵候选。

最优算法与最优矩阵如表 8.5 所示，并归纳总结在表 8.6 中。

表 8.5　SPS 的各类统计值(基于不同的 MAFFT 算法和替换矩阵)

方法	数据集	RV11		RV12		RV20		RV30		RV40		RV50	
	矩阵	max-SPS num	mean SPS	max-SPS num	mean SPS	max-SPS num	mean SPS	max-SPS num	mean SPS	max-SPS num	mean SPS	max-SPS num	mean SPS
L-NS-I	BLOSUM30	9	0.5132	19	0.8369	9	0.8532	14	0.7655	13	0.6688	5	0.7427
	BLOSUM45	12	0.5486	23	0.843	20	0.8577	12	0.7708	8	0.6777	8	0.7485
	BLOSUM62	15	0.545	17	0.838	26	0.8583	7	0.7686	6	0.6745	8	0.7466
	BLOSUM80	14	0.5449	14	0.8396	15	0.8573	12	0.7701	9	0.6752	7	0.7468
	PAM100	10	0.5279	12	0.8315	5	0.8535	10	0.7699	8	0.6686	3	0.7423
	PAM200	11	0.5244	13	0.8309	6	0.8518	5	0.7636	5	0.6682	0	0.7348
	sum		3.2041		5.0199		5.1318		4.6084		4.0329		4.4617
E-NS-I	BLOSUM30	6	0.499	15	0.8323	8	0.8508	12	0.7616	10	0.6669	3	0.7374
	BLOSUM45	12	0.5278	22	0.8414	21	0.8545	14	0.7705	10	0.6734	2	0.7425
	BLOSUM62	15	0.5338	16	0.836	18	0.8562	14	0.7695	4	0.6708	11	0.7469
	BLOSUM80	15	0.5254	19	0.8369	16	0.8564	11	0.7668	14	0.6729	8	0.7455
	PAM100	8	0.5124	14	0.829	8	0.852	4	0.7575	7	0.6665	5	0.7375
	PAM200	13	0.5155	11	0.8292	10	0.8501	5	0.7594	4	0.6647	2	0.7341
	sum		3.1138		5.0048		5.12		4.5852		4.0152		4.4439
G-NS-I	BLOSUM30	8	0.5201	17	0.8336	10	0.8527	12	0.7683	7	0.656	6	0.7384
	BLOSUM45	18	0.5803	24	0.8417	18	0.8565	15	0.7727	13	0.6623	10	0.7418
	BLOSUM62	14	0.5791	16	0.8388	19	0.8575	11	0.7783	12	0.6569	5	0.7432
	BLOSUM80	17	0.577	21	0.8397	13	0.8576	10	0.7737	13	0.6587	3	0.7425
	PAM100	11	0.5453	7	0.8286	12	0.8542	4	0.7694	3	0.6494	6	0.7401
	PAM200	9	0.5415	12	0.8288	11	0.852	8	0.7728	3	0.6518	1	0.7291
	sum		3.3432		5.0111		5.1306		4.6353		3.935		4.4351

表 8.6　最佳迭代算法与最佳替换矩阵

数据集	RV11	RV12	RV20	RV30	RV40	RV50
替换矩阵	BLOSUM45，BLOSUM80	BLOSUM30，BLOSUM45	BLOSUM45，BLOSUM62	BLOSUM30，BLOSUM45	BLOSUM30，BLOSUM80	BLOSUM45，BLOSUM62
迭代算法	G-NS-I	L-NS-I	L-NS-I	G-NS-I	L-NS-I	L-NS-I

2) 实验 2：最优参数 GOP/GEP/MATRIX 的确定

确定最优参数的思路如下：

(1) 根据表 8.6 的最佳算法与最佳替换矩阵(两个候选)，按照实验数据设置，得到每一组序列的 692×2 个 SPS 数据。

(2) 统计两个替换矩阵所对应的数据，通过比较 SPS 最大值，二者中选优者为该组序列的最佳矩阵。

(3) 定义两个参数：

① maxSPSnum 表示在该 GOP/GEP 下的 SPS 值=SPS 最大值的个数，意义是该值越大，则在该 GOP/GEP 下的可以达到最好比对效果的序列越多。例如，RV11 共 67 组序列簇，每一组有 692 个 GOP/GEP 组合，对应了 692 个 SPS 值，求出每一组的最大 SPS 值，即 RV11 共有 67 个 maxSPS。有 6 组序列在 GOP/GEP=2.9/0.6 的 SPS 值等于 maxSPS，则 maxSPSnum=6。

② meanSPS 表示在该 GOP/GEP 下 SPS 的平均值，意义是该值越大，则在该 GOP/GEP 下的整体比对效果越好。例如，RV11 共 67 组序列簇，每一组有 692 个 GOP/GEP 组合，即每个 GOP/GEP 组合有 67 个 SPS 值，当 GOP/GEP=2.9/0.6 时，将该组合下的 67 个 SPS 求和，再除以 67，可得平均值 meanSPS=0.5595。

(4) 根据这两个参数，得出 GOP/GEP 的候选值与对应的 meanSPS 和 maxSPSnum，统计在表 8.7 中。

(5) 综合考虑这两个参数，确定最优 GOP/GEP，如表 8.7 所示。

(6) 将所有的最优参数组合总结在表 8.8 中。

表 8.7　GOP/GEP 与对应的 meanSPS 和 maxSPSnum

数据集	矩阵	GOP	GEP	meanSPS	maxSPSnum
RV11	BLOSUM45	2.9	0.6	0.5595	6
			0.63	0.5577	6
			1.08	0.5432	6

(续表)

数据集	矩阵	GOP	GEP	meanSPS	maxSPSnum
RV11	BLOSUM45	2	0.12	0.5912	5
RV12	BLOSUM45	2.9	1.44	0.8465	15
		2.4	0.45	0.8512	10
RV20	BLOSUM62	2.3	0.63	0.8594	4
			0.69	0.859	4
			0.96	0.8567	4
			0.99	0.8567	4
			1.02	0.8565	4
			1.05	0.8565	4
			1.08	0.8565	4
			1.11	0.8566	4
			1.14	0.8565	4
		2.6	0.18	0.8617	2
RV30	BLOSUM45	2.1	0.72	0.7685	3
			0.75	0.7674	3
			0.78	0.766	3
RV30	Blosum45	2.4	0.15	0.7819	0
RV40	BLOSUM80	2.9	1.23	0.6518	2
			1.26	0.649	2
		2.8	0.39	0.6818	0

(续表)

数据集	矩阵	GOP	GEP	meanSPS	maxSPSnum
RV50	BLOSUM45	2.8	0.03	0.7455	2
			0.18	0.7443	2
			0.69	0.7377	2
			0.75	0.7379	2
		2.5	0.12	0.7505	0

表 8.8　最佳参数表

数据集	RV11	RV12	RV20	RV30	RV40	RV50
GOP	2	2.9	2.3	2.1	2.8	2.8
GEP	0.12	1.44	0.63	0.72	0.39	0.03
矩阵	BLOSUM45	BLOSUM45	BLOSUM62	BLOSUM45	BLOSUM80	BLOSUM45
算法	G-NS-I	L-NS-I	L-NS-I	G-NS-I	L-NS-I	L-NS-I

3) 实验 3：最优参数合理性的验证

图 8.8 给出了三种算法得出的数据集的 SPS 数值，mafft_measure 表示由表 8.8 归纳出的最佳参数得出的 SPS 分值，mafft_default 表示由 MAFFT 工具的默认参数(MAFFT-7.220-WIN64)得出的 SPS 分值，clustalw_default 表示由 CLUSTALW 工具的默认值(CLUSTALW-2.1-WIN)得出的 SPS 分数。表 8.9 统计了三种算法的 SPS 平均值，从图 8.8 和表 8.9 中可以得出如下结论：①MAFFT 默认值算出的 SPS 值大部分都高于 CLUSTALW 默认值的 SPS；②最优参数对应的 SPS 值高于 MAFFT 默认值的 SPS。说明表 8.8 得出的 MAFFT 工具最优参数的比对效果最好。因此，最优参数值优化了多序列比对的结果。

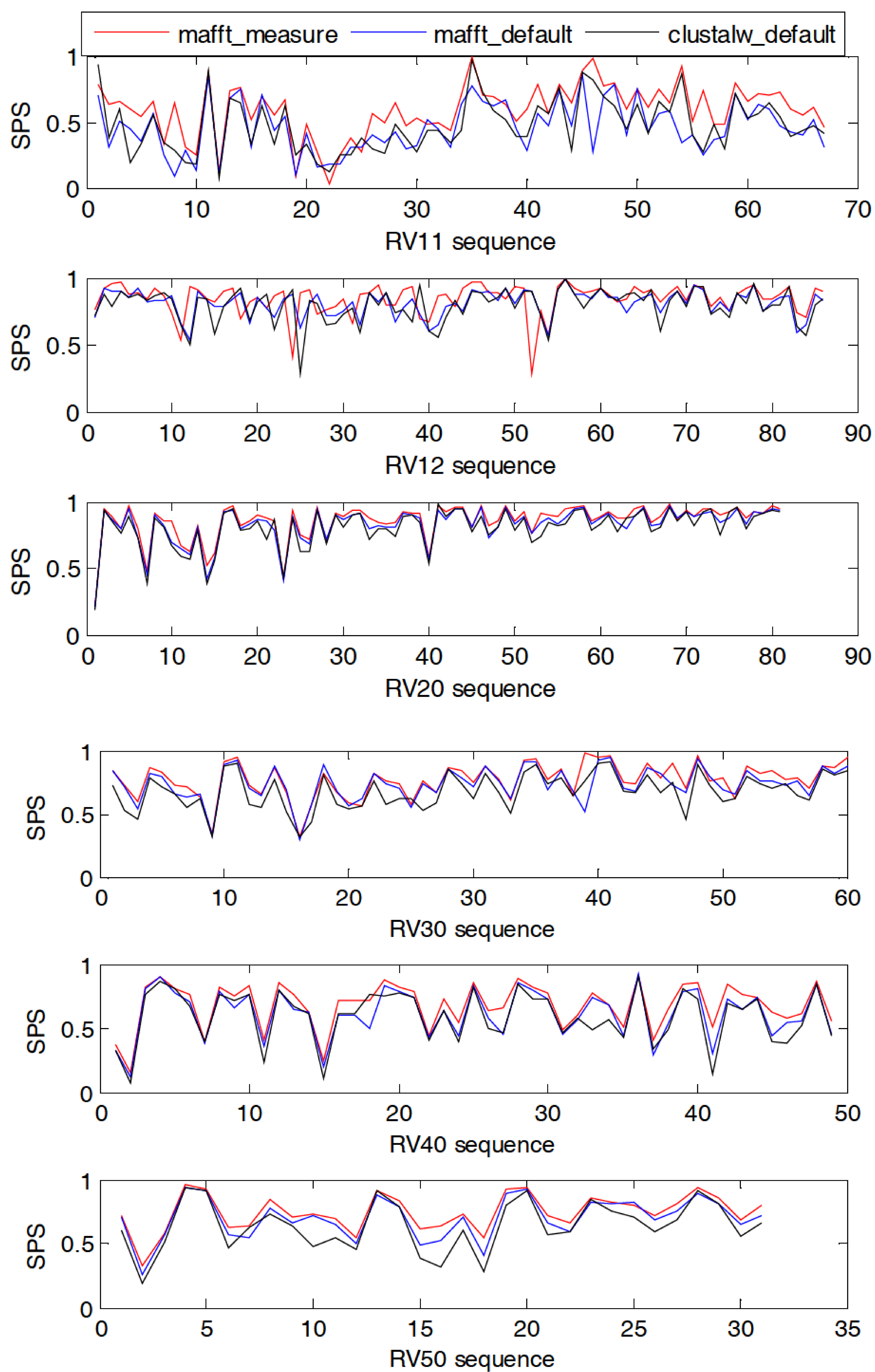


图 8.8　不同算法下的 SPS 值

表 8.9　最优参数的 SPS 与 MAFFT/CLUSTALW 默认值的 SPS 比较

数据集	RV11	RV12	RV20	RV30	RV40	RV50
FFT-NS-2 默认值-SPS 平均值	0.4582	0.8142	0.8301	0.737	0.6168	0.6971
L-NS-I 默认值-SPS 平均值	0.545	0.838	0.8583	0.7686	0.6745	0.7466
E-NS-I 默认值-SPS 平均值	0.5338	0.836	0.8562	0.7695	0.6708	0.7469
G-NS-I 默认值-SPS 平均值	0.5791	0.8388	0.8575	0.7783	0.6569	0.7432
CLUSTALW 默认值-SPS 平均值	0.4758	0.7966	0.8077	0.6802	0.5917	0.6377
CALC 计算值-SPS 平均值	**0.5912**	**0.8465**	**0.8594**	**0.7685**	**0.6818**	**0.7455**

8.2.4　结论

本节应用 MAFFT 工具解决多序列比对问题，为了得到更好的比对结果，在比对过程中摒弃常用的默认参数，致力于寻找最优的一组参数。应用 BAliBASE3.0 数据库实例从替换矩阵、空位罚分和与默认值横向比较这三个角度验证公式的合理性。实例证明最终的比对结果高度依赖于比对参数包括替换矩阵和空位罚分，一组好的参数会得到一个好的比对结果。实验结果表明，通过本节的最优参数，应用 MAFFT 工具，可以获得更高的精度校准，并获得更优质的比对结果。这项研究将优化多序列比对算法，并提供多序列比对的新思路。

在今后的工作中，还有一些需要完善的地方：①可以考虑应用

其他数据库的数据来测试本书公式的合理性与通用性。②可以应用本书思路寻找其他比对工具的最佳参数。注意，本书所得的最优参数都是基于 MAFFT 下产生的数据，因此该组参数仅适用于 MAFFT 比对工具，对于其他比对工具不一定通用。

8.3　本章小结

本章从两个角度研究多序列比对的参数问题。首先，基于 SP 目标函数，提出了明确的目标函数参数理论依据，给出替换矩阵的判断公式和空位罚分最佳取值公式，并应用 BAliBASE2.0 数据库实例从替换矩阵、空位罚分和与默认值横向比较这三个角度验证公式的合理性。实例证明最终的比对结果强烈依赖于 SP 目标函数的参数，包括替换矩阵和空位罚分，一组好的参数会得到一个好的比对结果。根据待测序列的长度条数相似度等信息代入相应公式可以得到最合适的起始空位罚分、延伸空位罚分和替换矩阵。实验结果表明，通过本章的公式，应用 MAFFT 工具，可以获得更高的精度校准，并获得更优质的比对结果。这项研究将优化多序列比对算法，并提供多序列比对的新思路。

另外，应用在线 MAFFT 工具进行多序列比对时，寻找最优的一组参数。应用 BAliBASE3.0 数据库实例从替换矩阵、空位罚分和与默认值横向比较这三个角度验证公式的合理性。实例证明最终的比对结果高度依赖于比对参数包括替换矩阵和空位罚分，一组好的参数会得到一个好的比对结果。实验结果表明，通过本文的最优参数，应用 MAFFT 工具，我们可以获得更高的精度校准，并获得更优质的比对结果。

参 考 文 献

[1] Lathrop R H. The Protein Threading Problem With Sequence Amino Acid Interaction Preferences Is Np-Complete[J]. Protein Eng, 1995, 7(9): 1059-1068.

[2] Wang L, Jiang T. On the complexity of multiple sequence alignment[J]. Journal of　Computational Biology, 1994, 1(2): 337-348.

[3] Ben Othman M T, Abdel-Azim G. Genetic algorithms with permutation coding for multiple sequence alignment[J]. Recent Patents on Dna& Gene Sequences, 2013, 7(2): 105-114.

[4] Notredame C. Recent progress in multiple sequence alignment: a survey[J]. Pharmacogenomics, 2002, 3(1): 131-144.

[5] Chuong B, Do, Katoh K. Protein multiple sequence alignment[M]. Functional Proteomics. Humana Press, 2008: 379-413.

[6] 张敏. 生物信息学中多序列比对等算法的研究[D]. 大连理工大学，2005.

[7] 邹权，郭茂祖，韩英鹏，等. 多序列比对算法的研究进展[J]. 生物信息学，2011, 08(4): 311-315.

[8] Gusfield D. Algorithms on strings, trees and sequences: computer science and computational biology[M]. Cambridge university press, 1997.

[9] Kaya M, Sarhan A, Alhajj R. Multiple sequence alignment with affine gap by using multi-objective genetic algorithm[J]. Computer methods and programs in biomedicine, 2014, 114(1): 38-49.

[10] Kwan M, Xiao N, Ding G. Assessing Activity Pattern Similarity with Multidimensional Sequence Alignment Based on a Multiobjective Optimization Evolutionary Algorithm[J]. Geographical Analysis, 2014, 46(3): 297-320.

[11] 张琎，张远. 基于 GC-GM 的多序列比对穷举遗传算法[J]. 计算机应用，2010，30(1)：146-149.

[12] Thompson J D, Higgins D G, Gibson T J. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice[J]. Nucleic acids research, 1994, 22(22): 4673-4680.

[13] Gondro C, Kinghorn BP. A simple genetic algorithm for multiple sequence alignment[J]. Genetics and Molecular Research, 2007, 6(4): 964-982.

[14] Dan D B, Kececioglu J. Parameter advising for multiple sequence alignment[J]. Bmc Bioinformatics, 2015, 16(1): 516-518.

[15] Notredame C, Higgins DG, Heringa J. T-COFFEE: a novel method for fast andaccurate multiple sequence alignments[J]. J.Mol.Evol., 2000, 302(1): 205-217.

[16] Katoh K, Toh H. Recent developments in the MAFFT multiple sequence alignment program[J]. Briefings in Bioinformatics, 2008, 9(4): 286-298.

[17] Pais F S, Ruy P C, Oliveira G, et al. Assessing the efficiency of multiple sequence alignment programs[J]. Algorithms for Molecular Biology Amb, 2014, 9(6): 78-87.

[18] Ahola V, Aittokallio T, Vihinen M, et al. A statistical score for assessing the quality of multiple sequence alignments[J]. Bmc Bioinformatics, 2006, 7(1): 152-169.

[19] http://mafft.cbrc.jp/alignment/software/eval/accuracy.html

[20] Murata M, Richardson J S, Sussman J L. Simultaneous comparison of three protein sequences[J]. Proceedings of the National Academy of Sciences, 1985, 82(10): 3073-3077.

[21] Gotoh O. Multiple sequence alignment: algorithms and applications[J]. Advances in Biophysics, 1999, 36(99): 159-206.

[22] Bahr A, Thompson J D, Thierry J, et al. BAliBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations[J]. Nucleic Acids Research, 2001, 29(1): 323-326.

[23] Thompson J D, Plewniak F, Poch O. BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs[J]. Bioinformatics, 1999, 15(1): 87-88.

[24] Kazutaka, Katoh, Kazuharu, Misawa, Kei-ichi, Kuma, et al. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform[J]. Nucleic Acid Res, 2002, 30(14): 3059-3066.

[25] Katoh K, Kuma K, Toh H, et al. MAFFT version 5: improvement in accuracy of multiple sequence alignment[J]. Nucleic Acids Research, 2005, 33(2): 511-518.

[26] Kazutaka K, Standley D M. MAFFT multiple sequence alignment software version 7: improvements in performance and usability.[J]. Molecular Biology & Evolution, 2013, 30(4): 102-343.

[27] Thompson Julie D, Koehl Patrice, Ripp Raymond, et al. BAliBASE 3.0: Latest developments of the multiple sequence alignment benchmark[J]. Proteins-structure Function & Bioinformatics, 2005, 61(1): 127-136.

附录　相关的源代码

附录 A　基本遗传算法总程序

```
clearall
tic;      %统计运行时间

%参数赋值
NUM_ge=1;     %全部大循环，计算几次
NUM_Individual=50;      %种群规模
eli_rate=0.1;       %保留精英个体 10%
cross_rate=0.6;     %交叉概率
mutation_rate=0.1;     %变异概率
unimprove=500;       %不变最大次数!
besttseq=cell(1,NUM_ge);
seqfit=cell(1,NUM_ge);
mseq=seqref('451c_ref1.msf');          %参考序列
se=load('sequence_bio_var_50.mat');
%从存储种群个体的文件中读取个体

%全体大循环，以便统计多次运算结果的最优值/平均值等数据
for seqi=1:NUM_ge
```

```
%计算所有个体的适应度值
for i=1:NUM_Individual
fitvalue(i)=fitness_affinedsp_spss_tcs(se.initpop{i},
mseq);
end

%赋初值
oldpop=se.initpop;
oldfitvalue=fitvalue;
  num_iteration=2000;    %最大迭代次数

%开始迭代
for gi=1:num_iteration

[newpop1,eli_Individual]=selection(oldpop,oldfitvalue,eli_
rate);    %选择操作
newpop2=struct_tsshoraffcrossover4to2(newpop1,cross_
rate,mseq);    %交叉操作
newpop3=struct_newaffmutation(newpop2,mutation_rate);
%变异操作
[rm,rn]=size(newpop3);
[em,en]=size(eli_Individual);

%更新种群
for i=1:NUM_Individual
if i<=rn
   newpop{i}=newpop3{i};
   %参与遗传操作的种群赋值给下一代
else
```

```
        newpop{i}=eli_Individual{i-rn};
       %精英直接复制到下一代
    end
    newfitvalue(i)=fitness_affinedsp_spss_tcs(newpop{i},
mseq);
    end

    oldfitvalue=newfitvalue;    %适应度值重新赋值迭代
    oldpop=newpop;    %种群重新赋值迭代
    bestfitvalue4to2(gi)=max(newfitvalue);  %输出最佳适应度值

    %当函数值 unimprove 次都不变，则收为收敛，跳出循环
    kb=0;
    if gi>=200    %至少迭代 200 次再考虑判断收敛
    if bestfitvalue4to2(gi)==bestfitvalue4to2(gi-1)
    kb=kb+1;
    else
    kb=0;
    end
    end
    if kb==unimprove            %设置跳出循环
    break
    end

    end                          %迭代结束



best_Individual=newpop{find(newfitvalue==max(newfitvalue))};
```

```
%找到最优个体

    %匹配列下注明*号
     [bm,bn]=size(best_Individual);
    bestsequence=char(bm+1,bn);
    bestsequence=best_Individual;

    for j=1:bn
    yu=find(bestsequence(:,j)==bestsequence(1,j));
    %寻找每一行和第一行相同的字母
    gu(j)=length(yu);             %统计当前列与第一行字母相同的数目
    end

    bestsequence(bm+1,find(gu==bm))='*';
    %如果当前列字母完全相同匹配，则在下方标柱*
    kx=length(find(bestsequence(bm+1,:)=='*'));


    %%%输出数据
      num_iter(seqi)=gi;            %输出收敛代数

    seqfit{seqi}=bestfitvalue4to2;
    %输出每一次循环的所有目标函数值(供画出收敛曲线)

    besttseq{seqi}=bestsequence;
    %输出有注匹配列为*的最佳个体
      ksx(seqi)=kx; %输出几个*
    bestsps(seqi)=SPS(best_Individual,mseq);
    besttcs(seqi)=TCS(best_Individual,mseq);
```

```
  T(seqi)=toc; %统计运行时间
time(1)=T(1);
if seqi>=2
time(seqi)=T(seqi)-T(seqi-1);
end

end                  %结束整个大循环

%存储所有相关数据
save('bestseq_451c_h_cm06_1st.mat','num_iter','besttseq',
'ksx','bestsps','besttcs','time','seqfit');
```

附录 B　生成初始种群 bio_var

```
%生物特性初始化，剩余空格随机插入
function
initpop=bioinitilize44(sequence_set,NUM_Individual)
a=char(sequence_set);
aa=cellstr(a);
maxlength=length(a(1,:));
[am,an]=size(a);
NUM_SEQ=am;              %定义比对序列的个数
s=cell(1,NUM_Individual);
%定义初始种群为细胞型数组，则每个元素个体可用字符矩阵表示

%定义每一行的空格数、空格倍数和剩余空格

for ii=1:NUM_Individual     %产生 NUM_Individual 个个体
```

```
    L=ceil(maxlength+0.2*maxlength*rand);
if L==maxlength
    L=ceil(maxlength*1.1);
end

b=ones(NUM_SEQ,L);   %定义字符矩阵维数
bb=char(b);
bbb=char(b);  %转化成字符矩阵
cc=cell(NUM_SEQ+1,L);

for i=1:NUM_SEQ
        len(i)=L-length(aa{i});    %每一行可插入的空格数
        gaplenth(i)=floor(len(i)*len(i)/L); %连续空格长度
if len(i)<=2&gaplenth(i)<2
gaplenth(i)=len(i);
elseif len(i)>2&gaplenth(i)<2
gaplenth(i)=2;
end
        gap_num(i)=floor(len(i)/gaplenth(i));
        %几个连续空格
        remainder_gap(i)=mod(len(i),gaplenth(i));
        %不够连续空格的剩余空格
        re(i)=gap_num(i)+1;   %剩余空格的位置

if gaplenth(i)==0        %如果没有连续空格，则随机插入剩余空格
in(1)=randint(1,1,[1,L-remainder_gap(i)+1]);
b(i,in(1):in(1)+remainder_gap-1)=0;
else%如果有连续空格
in(1)=randint(1,1,[1,floor(L/gap_num(i))-gaplenth(i)+
```

```
        1]);   %先定位第一个连续空格的位置
for j=2:gap_num(i)

in(j)=randint(1,1,[in(j-1)+gaplenth(i),j*floor(L/gap_num
        (i))-gaplenth(i)+1]);  %再定位后面所有的连续空格的位置
end
for j=1:gap_num(i)
b(i,in(j):in(j)+gaplenth(i)-1)=0;     %连续空格赋值为 0
end
cc{i}=findstr(b(i,:),1);     %找非空格的位置
if cc{i}~=0
 [mc,nc]=size(cc{i});
rr=randperm(nc);
br=cc{i}(rr(1:remainder_gap(i)));
b(i,br)=0;    %在非空格位随机插入剩余空格
end
end
end


%在每一行的非空格位置写入字符，空格位置写入“—”
m=0;
for i=1:NUM_SEQ
for j=1:L
if b(i,j)==1
m=m+1;
b(i,j)=a(i,m);
end
if b(i,j)==0
```

```
b(i,j)='_';
end
end
m=0;
end

bb=char(b);

bb2=bb;    %产生个体矩阵
bb=bbb;    %字符矩阵初始化进行循环
s{ii}=bb2;    %定义初始种群的个体为每一次插入空格的矩阵

%同一列都是空格的，删除此列
k=0;
for j=1:L
    c1=find(s{ii}(:,j)=='_');
if length(c1)~=NUM_SEQ    %如果此列不全是空格
    k=k+1;
    s2{ii}(:,k)=s{ii}(:,j);    %将此列赋值给 s2
end
end

end

initpop=s2;
```

附录 C　生成初始种群 rand_var

```
%变长度，一般初始化，随机插入空格
function
initpop=randinitilize(sequence_set,NUM_Individual)
%输入序列
a=char(sequence_set);
aa=cellstr(a);
maxlength=length(a(1,:));
[am,an]=size(a);
NUM_SEQ=am;            %定义比对序列的个数
s=cell(1,NUM_Individual);
%定义初始种群为细胞型数组，则每个元素个体可用字符矩阵表示

%定义每一行的空格数、空格倍数和剩余空格

for ii=1:NUM_Individual      %产生 NUM_Individual 个个体

   L=ceil(maxlength+0.2*maxlength*rand);
if L==maxlength
      L=ceil(maxlength*1.1);
end

   b=zeros(NUM_SEQ,L);   %定义字符矩阵维数
bb=char(b);
bbb=char(b);  %转化成字符矩阵
in_gap=zeros(NUM_SEQ,L);
for i=1:NUM_SEQ
```

```
        len(i)=L-length(aa{i});   %每一行可插入的空格数
    in_gap(i,:)=randperm(L);      %将空格的可插入位置随机打乱
    gap_innum=unique(in_gap(i,1:len(i)));
    %取前几个数作为插入空格的位置，并排序
    bb(i,gap_innum)='_';          %每一行相应随机位置替换成空格
    end

    %在每一行的非空格位置写入字符
    m=0;
    for i=1:NUM_SEQ
    for j=1:L
    if bb(i,j)==0
    m=m+1;
    bb(i,j)=a(i,m);
    end
    end
    m=0;
    end



    bb2=bb;   %产生个体矩阵
    bb=bbb;   %字符矩阵初始化进行循环
    s{ii}=bb2;   %定义初始种群的个体为每一次插入空格的矩阵

    %同一列都是空格的，删除此列
    k=0;
    for j=1:L
        c1=find(s{ii}(:,j)=='_');
```

```
if length(c1)~=NUM_SEQ    %如果此列不全是空格
    k=k+1;
    s2{ii}(:,k)=s{ii}(:,j);  %将此列赋值给 s2
end
end


end
initpop=s2;
```

附录 D　选择算子 selection

```
function
[newpop,eli_Individual]=selection(oldpop,fitvalue,eli_rate)
s=oldpop;
score33=fitvalue;    %读取个体适应度每个元素
score44=fitvalue;
 [sm,sn]=size(s);
NUM_Individual=sn;     %种群中个体的数目
%nnew=cell(1,NUM_Individual);
sc2=cell(1,NUM_Individual);
%选择最优的 10%作为下一代
ttt=-1*sort(-1*score33);     %从大到小排序后的个体适应度
elitism=round(eli_rate*NUM_Individual);  %精英数为最优的 10%
%scc=cell(1,elitism);
eli_Individual=cell(1,elitism);
new=cell(1,NUM_Individual-elitism);
```

```
%轮盘赌
ttk=cell(1,elitism);
[tm,tn]=size(ttt);
ttt(tn)=0;
k=0;
for j=1:tn-1
if ttt(j)~=ttt(j+1)
       k=k+1;
ttk{k}=find(score33==ttt(j));
end
end
[ttm,ttn]=size(ttk);
for i=1:ttn
    [m(i),n(i)]=size(ttk{i});
end
bn=cumsum(n);
an=zeros(1,tn);
for j=1:ttn
if j==1
an(1:bn(j))=ttk{j};
else
an(bn(j-1)+1:bn(j))=ttk{j};
end
end

    eli_Individual=s(an(1:elitism));  %精英个体
    s(an(1:elitism))=[];    %删除精英个体之后的种群

%eli_Individual=scc;
```

```
sc2=s;      %剩余种群
score44(an(1:elitism))=[];      %剩余种群的个体适应度
totalfit=sum(score44(1:NUM_Individual-elitism));
%求适应度总和

for i=1:NUM_Individual-elitism
   fitvalue(i)=score44(i)/totalfit;    %计算个体适应度概率
end


fitvalue1=cumsum(fitvalue);      %求累计概率
ms=sort(rand(1,NUM_Individual-elitism));
%产生剩余种群个体数目的[0,1]随机数
fitin=1;
newin=1;
while newin<=NUM_Individual-elitism
%循环剩余种群个体数目次
if ms(newin)<=fitvalue1(fitin)  %若随机数小于累计概率
   new{newin}=sc2{fitin};          %选择该位置的个体为下一代
   newin=newin+1;                  %判断下一个个体
else
   fitin=fitin+1;
%若随机数大于累计概率，则淘汰当前位置个体，选择下一个个体为下一代
end
end
newpop=new;
```

附录 E　横向多行交叉算子 hhor_crossover4to2

```
function
newpop=struct_tsshhoraffcrossover4to2(oldpop,cross_
rate,mseq)
new=oldpop;
new1=new(randperm(numel(new)));
%打乱选择后的个体排列顺序，尽量避免邻近两者相同

%[sm,sn]=size(new);
NUM_remain_indi=numel(new);     %种群中个体的数目
[spm,spn]=size(new{1});
NUM_SEQ=spm;                    %定义比对序列的个数

%如果参与交叉的个体个数是奇数，则调成偶数(将最后的个体从交叉候选
中删除)
cross_rnd=rand(1,NUM_remain_indi);
y1=find(cross_rnd<cross_rate);     %交叉候选
y2=find(cross_rnd>=cross_rate);
len=length(y1);
if len>2&mod(len,2)==1             %如果交叉个体数目是奇数
    y2(length(y2)+1)=y1(len);      %将最后的个体编号移至非候选中
    y1(len)=[];      %从交叉候选中删除最后的个体编号
end
num_cross=length(y1)/2;            %交叉配对的个数

 parent1=cell(1,num_cross);        %交叉的父代 1 初始化
 parent2=cell(1,num_cross);        %交叉的父代 2 初始化
```

```
 child1=cell(1,num_cross);       %交叉之后的子代 1 初始化
 child2=cell(1,num_cross);       %交叉之后的子代 2 初始化
%相邻两个进行配对(相当于随机两两配对)
if length(y1)>=2
for i=0:2:length(y1)-2
        j=i/2+1;                 %确定每一对的编号
parent1(j)=new1(y1(i+1));
parent2(j)=new1(y1(i+2));
end
end

child1=parent1;                    %先把父代 1 赋值给子代 1
child2=parent2;                    %先把父代 2 赋值给子代 2

for ii=1:num_cross                 %对所有交叉对的大循环
  nn1=length(parent1{ii});         %必须知道第一个父代的列数
  nn2=length(parent2{ii});         %必须知道第二个父代的列数
  r=unidrnd(NUM_SEQ);
  %从父体第一行到最后一行中随机选择可交叉的行位点

%若第一个父体比第二个父体长，在第二个父体末端补齐空格并交换行
if nn1>nn2
child2{ii}(:,nn2+1:nn1)='_';
     b1=child2{ii}(r:NUM_SEQ,:);
      b2=child1{ii}(r:NUM_SEQ,:);
child1{ii}(r:NUM_SEQ,:)=b1;
child2{ii}(r:NUM_SEQ,:)=b2;

%若第二个父体比第一个父体长，在第一个父体末端补齐空格并交换行
```

```
elseif nn2>nn1
child1{ii}(:,nn1+1:nn2)='_';
    b1=child2{ii}(r:NUM_SEQ,:);
      b2=child1{ii}(r:NUM_SEQ,:);
child1{ii}(r:NUM_SEQ,:)=b1;
child2{ii}(r:NUM_SEQ,:)=b2;

%若两个父体长度一样，直接交换行
elseif nn2==nn1
      b1=child2{ii}(r:NUM_SEQ,:);
      b2=child1{ii}(r:NUM_SEQ,:);
child1{ii}(r:NUM_SEQ,:)=b1;
child2{ii}(r:NUM_SEQ,:)=b2;
end

%同一列都是空格的，删除此列

cnn1(ii)=length(child1{ii});
cnn2(ii)=length(child2{ii});

   ck1=0; ck2=0;
for j=1:cnn1(ii)
    cc1=find(child1{ii}(:,j)=='_');
if length(cc1)~=NUM_SEQ                %如果此列不全是空格
    ck1=ck1+1;
    ch1{ii}(:,ck1)=child1{ii}(:,j);  %将此列赋值给ch1
end
end
```

```
for j=1:cnn2(ii)
    cc2=find(child2{ii}(:,j)=='_');
if length(cc2)~=NUM_SEQ
        ck2=ck2+1;
ch2{ii}(:,ck2)=child2{ii}(:,j);
end
end


%根据适应度值选择前两名作为子代
 pscore1=0;
pscore2=0;
cscore1=0;
cscore2=0;

ppscore1(ii)=fitness_affinedsp_spss_tcs(parent1{ii},
mseq);
ppscore2(ii)=fitness_affinedsp_spss_tcs(parent2{ii},
mseq);
ccscore1(ii)=fitness_affinedsp_spss_tcs(ch1{ii},mseq);
ccscore2(ii)=fitness_affinedsp_spss_tcs(ch2{ii},mseq);

%将父 12 和子 12 放在一起排序，取适应度前两名作为子 12
a(ii,:)=[ppscore1(ii),ppscore2(ii),ccscore1(ii),
ccscore2(ii)];
b(ii,:)=sort(a(ii,:));
c1=findstr(a(ii,:),b(ii,4));  %适应度第一名
c2=findstr(a(ii,:),b(ii,3));  %适应度第二名
if c1(1)==1     %有时会出现两个一样的最大值，则返回第一位元素
```

```
    newcross1{ii}=parent1{ii};   %将父 1 赋给子 1
elseif c1(1)==2
    newcross1{ii}=parent2{ii};   %将父 2 赋给子 1
elseif c1(1)==3
    newcross1{ii}=ch1{ii};       %将子 1 赋给子 1
elseif c1(1)==4
    newcross1{ii}=ch2{ii};       %将子 2 赋给子 1
end
if c2(1)==1
    newcross2{ii}=parent1{ii};   %将父 1 赋给子 2
elseif c2(1)==2
    newcross2{ii}=parent2{ii};   %将父 2 赋给子 2
elseif c2(1)==3
    newcross2{ii}=ch1{ii};       %将子 1 赋给子 2
elseif c2(1)==4
    newcross2{ii}=ch2{ii};       %将子 2 赋给子 2
end


end



%除复制 10%之外的所有个体
new00=new1(y2);
for i=1:NUM_remain_indi
if i<=num_cross
        new_se_cr{i}=newcross1{i};
elseif i>=num_cross+1&&i<=2*num_cross
```

```
        new_se_cr{i}=newcross2{i-num_cross};
    else
        new_se_cr{i}=new00{i-2*num_cross};
    end
    end
    newpop=new_se_cr;
```

附录 F　纵向交叉算子 ver_crossover4to2

```
    function
newpop=struct_tssveraffcrossover4to2(oldpop,cross_
rate,mseq)
    new=oldpop;
    new1=new(randperm(numel(new)));    %打乱选择后的个体排列顺
序，尽量避免邻近两者相同
    NUM_remain_indi=numel(new);    %种群中个体的数目
    [spm,spn]=size(new{1});
    NUM_SEQ=spm;              %定义比对序列的个数
    gap0=zeros(1,spn);   %定义足够多的空格
    gap=char(gap0);
    for i=1:spn
    gap(i)='_';
    end
    %如果参与交叉的个体个数是奇数，则调成偶数(将最后的个体从交叉候选
     中删除)
    cross_rnd=rand(1,NUM_remain_indi);
    y1=find(cross_rnd<cross_rate);    %交叉候选
    y2=find(cross_rnd>=cross_rate);
```

```
len=length(y1);
if len>2&mod(len,2)==1          %如果交叉个体数目是奇数
    y2(length(y2)+1)=y1(len); %将最后的个体编号移至非候选中
    y1(len)=[];      %从交叉候选中删除最后的个体编号
end
num_cross=length(y1)/2;       %交叉配对的个数

 parent1=cell(1,num_cross); %交叉的父代 1 初始化
 parent2=cell(1,num_cross); %交叉的父代 2 初始化
 child1=cell(1,num_cross);   %交叉之后的子代 1 初始化
 child2=cell(1,num_cross);   %交叉之后的子代 2 初始化
 stt1=cell(NUM_SEQ,1);
 %定义细胞型变量，方便存储，4 个序列参与比对
 stt2=cell(NUM_SEQ,1);
 %定义细胞型变量，方便存储，4 个序列参与比对


%相邻两个进行配对(相当于随机两两配对)
if length(y1)>=2
for i=0:2:length(y1)-2
        j=i/2+1;  %确定每一对的编号
parent1(j)=new1(y1(i+1));
parent2(j)=new1(y1(i+2));
end
end


%对所有交叉对的大循环
for ii=1:num_cross
```

```
  nn1=length(parent1{ii});      %必须知道第一个父代的列数
  nn2=length(parent2{ii});      %必须知道第二个父代的列数

for i=1:NUM_SEQ
stt1{i}=find(parent1{ii}(i,:)~='_');
%寻找第一个父代非空格字符的位置
n1(i)=length(stt1{i});        %确认第一个父代的尺寸
stt2{i}=find(parent2{ii}(i,:)~='_');
%寻找第二个父代非空格字符的位置
end

minn=min(n1)-1;
%寻找最后可交叉的位置，不得处于最短序列的最后一个非空格字符
r=unidrnd(minn);
%从第一个到最后一个字符中随机选择可交叉的位点


for i=1:NUM_SEQ

len1(i)=length(strcat(parent1{ii}(i,1:stt1{i}(r)),
parent2{ii}(i,stt2{i}(r)+1:nn2)));
%交叉之后不等长，求每行的长度

len2(i)=length(strcat(parent2{ii}(i,1:stt2{i}(r)),
parent1{ii}(i,stt1{i}(r)+1:nn1)));
end

maxlen1=max(len1);          %最长行
```

```
maxlen2=max(len2);

for i=1:NUM_SEQ
gaplen1(i)=maxlen1-len1(i);     %插入空格的个数
gaplen2(i)=maxlen2-len2(i);
if gaplen1(i)>0

child1{ii}(i,:)=strcat(parent1{ii}(i,1:stt1{i}(r)),
gap(1:gaplen1(i)),parent2{ii}(i,stt2{i}(r)+1:nn2));
%在交叉处补齐空格直至等长
else
child1{ii}(i,:)=strcat(parent1{ii}(i,1:stt1{i}(r)),
parent2{ii}(i,stt2{i}(r)+1:nn2));
%最长行不用插入空格
end
if gaplen2(i)>0

child2{ii}(i,:)=strcat(parent2{ii}(i,1:stt2{i}(r)),
gap(1:gaplen2(i)),parent1{ii}(i,stt1{i}(r)+1:nn1));
else

child2{ii}(i,:)=strcat(parent2{ii}(i,1:stt2{i}(r)),
parent1{ii}(i,stt1{i}(r)+1:nn1));
%最长行不用插入空格
end
end


%同一列都是空格的，删除此列
```

```
cnn1(ii)=length(child1{ii});
cnn2(ii)=length(child2{ii});

   ck1=0; ck2=0;
for j=1:cnn1(ii)
    cc1=find(child1{ii}(:,j)=='_');
if length(cc1)~=NUM_SEQ    %如果此列不全是空格
    ck1=ck1+1;
    ch1{ii}(:,ck1)=child1{ii}(:,j);  %将此列赋值给 ch1
end
end

for j=1:cnn2(ii)
    cc2=find(child2{ii}(:,j)=='_');
if length(cc2)~=NUM_SEQ
       ck2=ck2+1;
ch2{ii}(:,ck2)=child2{ii}(:,j);
end
end


%根据适应度值选择前两名作为子代
 pscore1=0;
pscore2=0;
cscore1=0;
cscore2=0;
ppscore1(ii)=fitness_affinedsp_spss_tcs(parent1{ii},
mseq);
```

```
    ppscore2(ii)=fitness_affinedsp_spss_tcs(parent2{ii},
mseq);
    ccscore1(ii)=fitness_affinedsp_spss_tcs(ch1{ii},mseq);
    ccscore2(ii)=fitness_affinedsp_spss_tcs(ch2{ii},mseq);

    %将父 12 和子 12 放在一起排序，取适应度前两名作为子 12
    a(ii,:)=[ppscore1(ii),ppscore2(ii),ccscore1(ii),
    ccscore2(ii)];
    b(ii,:)=sort(a(ii,:));
    c1=findstr(a(ii,:),b(ii,4));  %适应度第一名
    c2=findstr(a(ii,:),b(ii,3));  %适应度第二名
    if c1(1)==1        %有时会出现两个一样的最大值，则返回第一位元素
        newcross1{ii}=parent1{ii};   %将父 1 赋给子 1
    elseif c1(1)==2
        newcross1{ii}=parent2{ii};   %将父 2 赋给子 1
    elseif c1(1)==3
        newcross1{ii}=ch1{ii};       %将子 1 赋给子 1
    elseif c1(1)==4
        newcross1{ii}=ch2{ii};       %将子 2 赋给子 1
    end
    if c2(1)==1
        newcross2{ii}=parent1{ii};   %将父 1 赋给子 2
    elseif c2(1)==2
        newcross2{ii}=parent2{ii};   %将父 2 赋给子 2
    elseif c2(1)==3
        newcross2{ii}=ch1{ii};       %将子 1 赋给子 2
    elseif c2(1)==4
        newcross2{ii}=ch2{ii};       %将子 2 赋给子 2
    end
```

```
end

%除复制10%之外的所有个体
new00=new1(y2);
for i=1:NUM_remain_indi
if i<=num_cross
       new_se_cr{i}=newcross1{i};
elseif i>=num_cross+1&&i<=2*num_cross
       new_se_cr{i}=newcross2{i-num_cross};
else
       new_se_cr{i}=new00{i-2*num_cross};
end
end
newpop=new_se_cr;
```

附录 G　变异算子 mutation

```
%单点变异

function
newpop=struct_newaffmutation(oldpop,mutation_rate,mseq)
new_mu=oldpop;
%变异算子
%new_mu=initpop;
%mutation_rate=0.1;  %变异概率
%[sm,sn]=size(new_se_cr);
[nsm,nsn]=size(new_mu{1});
```

```
NUM_SEQ=nsm;
NUM_remain_indi=numel(new_mu);     %种群中个体的数目
mr=rand(1,NUM_remain_indi);
my1=find(mr<=mutation_rate);
%从种群中随机挑选变异的个体，确定其所在的位置
my2=find(mr>mutation_rate);
mnew00=new_mu(my2);    %未变异的个体
muparent=cell(1,length(my1));
for mi=1:length(my1)
    muparent{mi}=new_mu{my1(mi)};   %挑选参与变异的父代
end
inpop=muparent;

for i=1:numel(inpop)
mu_r=unidrnd(NUM_SEQ,1,1);     %随机选一行

 init=inpop{i}(mu_r,find(inpop{i}(mu_r,:)~='_'));
 %残基字符
 ingap=inpop{i}(mu_r,find(inpop{i}(mu_r,:)=='_'));
 %空格
 seq=find(inpop{i}(mu_r,:)~='_');      %残基字符的位置
 gap=find(inpop{i}(mu_r,:)=='_');      %空格位置
if length(gap)~=0
rs=unidrnd(length(seq),1,1);
rg=unidrnd(length(gap),1,1);
if rs>=1&rs<=length(seq)
 randseq=seq(rs);   %随机选择一个非空格(字符)位置
else
randseq=seq(1);
```

```
end
if rg>=1&rg<=length(gap)
 randgap=gap(rg);   %随机选择一个空格位置
else
randgap=gap(1);
end
  seq(find(seq==randseq))=randgap;
  %将选择的空格位置替换非空格位置
 seq=sort(seq);         %排序
 gap(find(gap==randgap))=randseq;
 %将选择的非空格位置替换空格位置
 gap=sort(gap);        %排序
  inpop{i}(mu_r,gap)=ingap;    %将空格读入
  inpop{i}(mu_r,seq)=init;     %将字符读入

end
%同一列都是空格的，删除此列
k=0;
for j=1:length(inpop{i})
    c1=find(inpop{i}(:,j)=='_');
if length(c1)~=NUM_SEQ   %如果此列不全是空格
    k=k+1;
    muchild{i}(:,k)=inpop{i}(:,j);  %将此列赋值给 muchild
end
end

%计算变异子代与父代的适应度值，挑选优的作为下一代
ppmscore(i)=fitness_affinedsp_spss_tcs(muparent{i},
mseq);
```

```
ccmscore(i)=fitness_affinedsp_spss_tcs(muchild{i},
mseq);
if ppmscore(i)>=ccmscore(i)
newmu{i}=muparent{i};
else
newmu{i}=muchild{i};
end


end

%除复制10%之外的所有个体
for i=1:NUM_remain_indi
if i<=length(my1)
newpop2{i}=newmu{i};
elseif i>=length(my1)+1&&i<=length(my1)+length(my2)
newpop2{i}=mnew00{i-length(my1)};
end
end
newpop=newpop2;
```

附录 H　适应度函数：SP 函数

```
%sum of pair 值，即测试序列中每一列残基在参考序列中两两匹配的分数

function sopi=SOP(testseq,msfseq)
ms=msfseq; %参考比对,空位用-表示，残基是大写字母
as=testseq;  %测试比对，空位用-表示，残基是大写字母
```

```
[am,an]=size(as);
[mm,mn]=size(ms);
c=cell(am,an);
a=cell(am,an);


for i=1:an
for j=1:am
if as(j,i)~='_'
    c{j,i}=findstr(as(j,i),ms(j,:));
    %测试序列残基在参考序列中的位置,同一行相同残基有多个位置
    a{j,i}=findstr(as(j,i),as(j,:));
    %测试序列残基在自身的位置
end
end
end
cc=c;

%按照先后顺序，将测试序列残基在参考序列的位置逐一安放,位置矩阵为
 cc(am,an)

for j=1:am
for i=1:an
if length(a{j,i})~=0
for kl=1:length(a{j,i})
cc{j,a{j,i}(kl)}=c{j,a{j,i}(length(a{j,i}))}(kl);
end
end
end
```

```
end


%如果测试列的残基和参考列的残基在同一列，则 cc(j)=cc(k)，p=p+1
p=0;
for i=1:an
for j=1:am-1
for k=j+1:am
if length(cc{j,i})~=0
if cc{j,i}==cc{k,i}
   p=p+1;
end
end
end
end
end

sopi=p;
```

附录 I　多序列比对参数研究的相关程序

Mafft 的批处理程序：

```
#!/usr/bin/perl
my $mafft = "D:/备份/D/work1/msa/test3/mafft.bat";
%mafft 的安装路径
@files=<*.tfa>;

foreach $file(@files){
```

```
    open F, $file or die $!;
my $aln=$file;   %需要输入的序列文件，多个序列必须在一个文件中
$file =~s/\.\w+//g;  %去掉扩展名
for(my $j = 1 ; $j <= 20; $j = $j + 1){
$jm=$j*0.1+0.01;
for(my $k = 0; $k <$j/2-0.01; $k = $k + 0.2) {
my $km=$k+0.1;
my $out="$file\_$jm\_$km.fasta";
my $system_check=system("$mafft --op $j --ep $k --bl 30
$aln>$out");
}
}
}
```

2 Clustalw 的批处理程序：

```
#!/usr/bin/perl
my $clustalw = "D:/备份/D/work1/msa/balibase3 数据库/cw/
ClustalW2";
%clustalw 的安装路径
@files=<*.india>;
foreach $file(@files){
    open F, $file or die $!;
    my $aln=$file; %需要输入的序列文件，多个序列必须在一个文件中
    $file =~s/\.\w+//g;#去掉扩展名。
    my $out="$file.fasta";
    #my $out="$file";
    my $system_check=system("$clustalw -OUTORDER=INPUT
    -OUTPUT=FASTA -INFILE=$aln -OUTFILE=$out");
```

```
    %实现比对,具体参数设置可参考 clustalw 的帮助文件中的 help 9
}
```

附录 J　HMM 和 QPSO 算法用于多序列比对的程序

```
function [fgbest,seq,T]=bqpsoxin(MAXITER)
tic;
sequence=fastaread('kinase_ref1.india');
lengthdata=length(sequence);
a=zeros(1,lengthdata);
for i=1:lengthdata
    a(1,i)=length(sequence(i).Sequence);
end
lengthmax=max(a);
popsizelength= ceil(lengthmax + 0.2*lengthmax * rand);
popsize=50;
f_gbest=0;
f_x=zeros(1,popsize);
f_pbest=zeros(1,popsize);
for i=1:popsize
 partical(i).x=ones(lengthdata,popsizelength);
 partical(i).seq=setstr(zeros(lengthdata,popsizelength));
 for
 j=1:lengthdatagap=randint(1,(popsizelength-length
(sequence(j).Sequence)),[0,length(sequence(j).Sequence)]);
 gap=sort(gap);
```

```
 partical(i).x(j,(1:length(gap)) + gap)=0;
 partical(i).seq(j,(1:length(gap)) + gap)='.';
 partical(i).seq(j,find(partical(i).seq(j,:)~='.'))=s
 equence(j).Sequence;
end
partical(i).pbest=partical(i).x;
 f_x(i)=SPSXIN2(partical(i).seq);
 f_pbest(i)=f_x(i);
end
[f_gbest,g]=max(f_pbest);
gbest=partical(g).pbest;
for t=1:MAXITER
%beta=0.7;
 beta=(1-0.5)*(MAXITER-t)/MAXITER+0.5;
    mbest=ones(lengthdata,popsizelength);
    ave=zeros(lengthdata,popsizelength);
for i=1:popsize
    ave=ave+partical(i).pbest;
end
    ave=ave./popsize;
for i1=1:lengthdata
    mbest(i1,find(ave(i1,:)>0.5))=1;
    mbest(i1,find(ave(i1,:)<0.5))=0;
    k1=find(ave(i1,:)==0.5);
for i2=1:length(k1)
if rand>0.5
    mbest(i1,k1(i2))=1;
else
    mbest(i1,k1(i2))=0;
```

```
end
end
end
for i=1:popsize
    partical(i).c=zeros(lengthdata,popsizelength);
for d=1:lengthdata
     partical(i).c(d,:)=ones(1,popsizelength);
     A=rand(1,popsizelength);
     partical(i).p(d,:)=A.*partical(i).pbest(d,:)+
     (1-A).*gbest(d,:);
     partical(i).p(d,find(partical(i).p(d,:)==0))=0;
     partical(i).p(d,find(partical(i).p(d,:)==1))=1;
     l1=find(partical(i).p(d,:)~=0 & partical(i). p(d,:)
     ~=1);
for r=1:length(l1)
if rand>0.5
     partical(i).p(d,l1(r))=1;
else
     partical(i).p(d,l1(r))=0;
end
end
if(rand<0.5)
     cpoint=round(rand*popsizelength);
     z1=[partical(i).pbest(d,1:cpoint),gbest(d,cpoint
     +1:popsizelength)];
     z2=[gbest(d,1:cpoint),partical(i).pbest(d,cpoint
     +1:popsizelength)];
if rand<0.5
     partical(i).p(d,:)=z1;
```

```
else
      partical(i).p(d,:)=z2;
end
      partical(i).c(d,:)=partical(i).x(d,:)-mbest(d,:);
      u=-log(rand);
      u2=find(partical(i).c(d,:)~=0);
      b=round(beta*length(u2)*u);
if b/popsizelength>1
        pr=1;
else
        pr=b/popsizelength;
end
for n=1:popsizelength
if rand<pr
if partical(i).p(d,n)==1
       partical(i).p(d,n)=0;
else
       partical(i).p(d,n)=1;
end
end
end
      partical(i).x(d,:)=partical(i).p(d,:);
      w1=find(partical(i).x(d,:)==0);
      v1=find(partical(i).x(d,:)==1);
if
      length(w1)>(popsizelength-length(sequence(d).Seq
      uence))
      num1=length(w1)-(popsizelength-length(sequence(d
      ).Sequence));
```

```
        num=randperm(length(w1));
        partical(i).x(d,w1(num(1:num1)))=1;
end
if length(w1)<(popsizelength-length(sequence(d).
Sequence))
        num1=(popsizelength-length(sequence(d).Sequence))
        -length(w1);
        num=randperm(length(v1));
        partical(i).x(d,v1(num(1:num1)))=0;
end
        w2=find(partical(i).x(d,:)==0);
        v2=find(partical(i).x(d,:)==1);
        partical(i).seq(d,w2)='.';
        partical(i).seq(d,v2)=sequence(d).Sequence;
end
%f_x(i)=sps(partical(i).seq)/ref1();
    f_x(i)=SPSXIN2(partical(i).seq);
if f_pbest(i)<f_x(i)
    f_pbest(i)=f_x(i);%f_pbest(i)
    partical(i).pbest=partical(i).x;%partical(i).pbest
end
end
          [f_gbest,g]=max(f_pbest);
          gbest=partical(g).pbest;
          fgbest(t)=f_gbest;
end
for r=1:lengthdata
    w3=find(gbest(r,:)==0);
    v3=find(gbest(r,:)==1);
```

```
    seq(r,w3)='.';
    seq(r,v3)=sequence(r).Sequence;
end
T=toc;
```